Modeling and Solving Resource Constrained Project Scheduling Problems with Remanufacturing Activities

Tiansheng Zhang

A Thesis

in

The Department

of

Mechanical, Industrial and Aerospace Engineering (MIAE)

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Industrial Engineering) at

Concordia University

Montréal, Québec, Canada

April 2021

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Tiansheng Zhang**
Entitled:	**Modeling and Solving Resource Constrained Project Scheduling Problems with Remanufacturing Activities**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair
Dr. Dr. Hossein Hashemi Doulabi

______________________________ External Examiner
Dr. Dr. Osama Moselhi

______________________________ Examiner
Dr. Name of Examiner One

______________________________ Supervisor
Dr. Mingyuan Chen

Approved by ______________________________
Martin D. Pugh, Chair
Department of Mechanical, Industrial and Aerospace Engineering (MIAE)

______________ 2021 ______________________________
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Modeling and Solving Resource Constrained Project Scheduling Problems with Remanufacturing Activities

Tiansheng Zhang

Resource constrained project scheduling problem (RCPSP) is one of the most important problems in industrial engineering and production management. Owing to environmental concerns, companies are paying more attention to the remanufacturing of end-of-life products. In this thesis, a mathematical model is developed considering remanufacturing activities in resource constrained project scheduling problem. The mathematical model considers recycle rate in multiple operation modes and several components of cost, including bonus, penalty, and others. A set of project network instance are generated using RanGen1 for evaluation. To solve the model, a three-stage heuristic method is developed in CPLEX 12.8 environment. Result shows that proposed method can reach a close-to-optimal solution within acceptable time limit.

Acknowledgement

First of all, I want to thank my supervisor Dr. Mingyuan Chen. 2020 is a really hard year for me. Without Dr. Chen’s help and patient guidance, I would not be able to finish my thesis. It’s a great experience to work with him.

I would like to express my gratitude to my office fellows Cesar, Omar, Yasser, Jaeir, Walaa, Zenan and Ruo, for all the great advice and experience. It’s a pleasure to work in the office with all of you.

I would like to thank my friends Neo, Ruozhu, Bo, Xingge, Xiangping and Giordano for their support and knowledge you shared with me.

Also, I want to thank Ontario Northland company for technical support.

Most importantly, I want to express my gratitude for my beloved family. A lot happened during my study, and they have been my best support to overcome every difficulty and finally reach the destination.

Table of Contents

List of Tables and Figures

Chapter 1 Introduction

In this chapter, we introduce the general information of resource constrained project scheduling problem (RCPSP) and remanufacturing system. The challenges are discussed with the outline of the thesis provided at the end of this chapter.

1.1 Resource constrained project scheduling

Since the structure of supply chain are getting more and more complex in recent decades, modern companies are gradually reorganizing themselves into a modular and project-oriented structure to avoid possible impact generated by activities delay. According to Bounds, project management is estimated to be an 850 million industry and is expected to grow by 20 percent each year [2]. Though many projects are started each year, only 26% of them are finished on time and within budget limit, which makes project management a more important subject [2].

Project scheduling originates from project management. In 1950s quantitative methods such as Critical Path Method (CPM) and Project Evaluation Review Techniques (PERT) are proposed to manage projects. These approaches give good estimation on project life circle if no resource conflict appears, which often is not the case. With the requirement of resource allocation appears, project management problems are more difficult to solve.

Resource constrained project scheduling problem (RCPSP) is proposed in late 1970s by

Pritsker et al. [53]. It is used to reach an optimal objective with respect to scarce resources. With different variants and extensions, RCPSP may reach various result with different constraints.

1.2 Remanufacturing system

The definition of remanufacturing is raised when build a single item which requires tremendous amount of resource to build. During World War II large scale project remanufacturing was put on high importance since productivity was in severe shortage in both civil and military fields. Cars need to be kept running and weapons need to be recycled and refurbished to satisfy the fast consumption.

Remanufacturing process aims to bring end-of-life assembly product back to like-new status with repairing and replacing components [3]. Comparing to traditional manufacturing, remanufacturing provides high-quality products with lower energy and resource consumption, while the profit margin is higher.

For remanufacturing enterprises, the challenge is to deal with highly varied production line caused by difference quality of recycled products. Disassembled components are inspected and stored for further reassembly. Furthermore, recycled components from a same batch may take different time to get prepared, makes it difficult to schedule the reuse.

One of the difficulties in remanufacturing enterprise is to create a disassembly schedule with consideration of recycled item usage and holding cost. An early remanufacturing activity may lead to unnecessary inventory stockpile while a late activity may result in delay on activities that requires refurbished item and lateness of whole project. Including remanufacturing schedule in a resource-constrained project scheduling problem can effectively decrease possibility of project delay.

1.3 Motivations and Challenges

Both RCPSP and remanufacturing have been thoroughly researched in the past several decades, respectively. However, research on the impact of remanufacturing on RCPSP is not rich. RCPSP focus on reasonable resource allocation over time to achieve a specific goal with known number of resources. For large scale project dealing with single or small-batch items, remanufacturing of several core parts can greatly save costs and shorten projects durations. But remanufacturing activities and RCPSP are usually considered separately since remanufacturing activities produce non-renewable resources instead of consumption.

The way remanufacturing activities do is to produce non-renewable resources while use renewable resources. Normal non-renewable resources are acquired from external source and does not need to be processed by workers and machines. Costs related to them are order placement cost, material purchase cost and inventory holding cost. For remanufacturing activities, resources need to be preprocessed before used by other activities. Because remanufacturing activities has a disassembly hierarchy and produce non-renewable resource which need additional cost for holding, remanufacturing activities should be considered separately in RCPSP without losing its characteristics.

Also, for refurbishment project, material requirement is not known until further inspection on items and non-renewable resources need to be ordered after project initiates. This can add additional complexity to problem solving. Besides, recycled materials may take different time to be prepared. Normal remanufacturing scheduling only focus on item disassembly hierarchy, which may lead to a local optimal for whole project. Integrating remanufacturing activities into resource-constrained project scheduling problem gives a more comprehensive consideration on refurbishment project scheduling.

1.4 Contribution

This research proposed a mathematical model for multi-mode resource constrained project scheduling project scheduling problem with remanufacturing activity and cumulative resources. The model integrates remanufacturing activity and cumulative resource as new components for RCPSP, which reduce the uncertainty of remanufacturing project and increase efficiency. Remanufacturing activities are considered in resource constrained project scheduling problem with the production and consumption of cumulative resource in multiple operation modes for the first time.

A three-stage heuristic method is proposed to quickly reach a suboptimal solution with acceptable accuracy in practical situation. 18 RanGen1 generated instance are used to verify the efficiency of proposed heuristic method, and the results are compared with solution generated by CPLEX embedded solver.

1.5 Outline

In the next chapter, literatures researching resource-constrained project scheduling problem (RCPSP) and remanufacturing scheduling will be reviewed. A mathematical model for RCPSP with remanufacturing activities will be demonstrated in Chapter 3. In Chapter 4, a three-stage heuristic method to solve problem will be presented. Chapter 5 gives a summary and future prospection on this study.

CHAPTER 2: Literature Review

2.1 Introduction

In recent decades, rapid progress has been seen in the field of project scheduling. Mathematical basis has been consolidated. Various methods are proposed for solving this NP-complete problem. Extensions and variants considering practical situations are introduced. In this chapter, a literature review related to resource-constrained project scheduling problem is given.

2.2 Project Management Tools

Project management can date back to at least 4000 years ago when ancient architectures projects are organized without modern methods and tools but only estimations. Before resource constrained project scheduling is proposed, tools like Gantt Chart and CPM are widely used for scheduling. Resource constraints are usually considered separate to project schedule.

2.2.1 Gantt Chart

Gantt chart, also known as harmonogram, is invented by Henry Gantt to demonstrate project schedules. A Gantt chart lists all tasks to be complete on vertical axis and time intervals on horizontal axis [4]. Gantt chart can clearly demonstrate start-end relationship for all tasks and non-complete work breakdown structure, but not enough to show precedence network for whole project. Therefore, tracing the impact of delay on whole project is hard on large scale project.

2.2.2 Critical Path Method

Critical Path Method (CPM) is an algorithm developed by Morgan R. Walker and James E. Kelley for scheduling a set of activities [5]. It uses four components to construct a model: a set of activities organized in work breakdown structure (WBS) directed by project aim; durations needed to complete activities; precedence relationship of activities; logical points indicating start and end, also known as milestones.

Critical path method generates a critical path through activities network which has the longest path. Other paths from start node to sink node that have shorter durations are called sub-critical path or non-critical path. A critical path determines minimum time needed to complete project if no task on this path is delayed, while some activities on non-critical path can be postponed without delay the whole project. The activities on critical path are supposed to be started once their precedence tasks are completed such that the project can be finished on earliest possible time.

Critical path method gives a good estimate on project makespan and provides earliest start time and latest start time for all tasks, which makes it possible for project manager to allocate additional resources to crash project duration and transform the project into a time-cost tradeoff problem. By shorten duration of tasks on critical path, the critical path may change, and non-critical path may become critical path, which can be further altered for shorter project makespan.

2.2.3 Project Evaluation and Review Technique

The Project Evaluation and Review Technique (PERT) is a statistical tool for project management developed by The US Navy. It is applied increasingly to Critical Path Method despite their difference [6].

PERT makes it possible to schedule a project without knowing the accurate duration of its tasks by considering uncertainty and provides estimations on project. Usually, four types of time are required for PERT scheduling: optimistic time, pessimistic time, most likely time and expected time, representing the minimum time required, the maximum time required, the best estimate when activity goes normal and the average time if activity is repeated under the same conditions.

PERT is often used with CPM for better project management. With different duration estimation Given by PERT, project managers are able to allocate more cash and resources into activities more likely to be critical, thus reduce the uncertainty of project.

2.3 Resource constrained project scheduling problem

Project duration is one of the most important factors when implementing a project. Improper managing a project may cause late completion of the project, which reduce possible profit. Many researchers have proposed methods to minimize makespan of a project. For multimode project scheduling, due to the introduction of non-renewable resource, minimizing overall project cost is considered as an important objective function. Among the research in RCPSP problem, many put their focus on easy-accessible general-purpose tools for integer programming problem and mixed-integer problem, which can be easily understood and deployed by practitioners without excessive background knowledge.

Kolisch [7] proposed a new RSM-based priority rule for parallel scheduling scheme. The concept of Generally Forbidden Pairs, Temporarily Forbidden Pairs and Current Schedulable Pairs are introduced. The earliest schedulable time $\prod_{(i,\,j)}^{r}$ considering resource r in Decision Set D_n at stage n for any activity pair $(i,\,j)$ can be calculated. A priority value $v(i)$ can be obtained with latest start time and $E_{(i,\,j)}$, the earliest time to schedule activitiy j if activity i is scheduled at time t_n. IRSM (Improved RSM) priority rule choose the activity with smallest $v(i)$ to schedule at each stage.

Hartmann [8] proposed a precedence feasible based permutation genetic algorithm for RCPSP. An activity sequence is denoted as an individual $I = (j_1^I, j_2^I,\ \dots, j_J^I)$, in which j_n^I denotes the nth scheduled activity. An activity sequence corresponds uniquely to a schedule with serial scheduling scheme. Two new sequences are created from two old sequences using crossover and mutation, then given an individual fitness. Four type of selection operators are used to choose an individual with highest fitness. This permutation-based GA method achieves better optimal value with less CPU time compared to other Genetic algorithm.

In Alcaraz and Maroto [10], a robust genetic algorithm has been proposed for resource allocation in project scheduling. Activity list representation is used with an additional gene called forward-backward gene denoting the mode this sequence is constructed. Three types of crossover are suggested, including Precedence set crossover, Forward-backward crossover (FBC) and Two-point forward-backward crossover (TP-FBC). The crossover keeps relative position of selected gene from one parent on its scheduling direction. Then mutation operator is applied after crossover and alters a gene position. FBC and TP-FBC gives good max deviation and optimal in ProGen-generated instance.

Merkle, Middendorf and Schemeck [11] introduce an ant colony optimization (ACO) on RCPSP with both serial and parallel schedule generation scheme are used for sequencing. The ACO algorithm use heuristic information η_{ij} and pheromone information τ_{ij} to denote how good it is to put activity j at place i. The probability of schedule activity j at place i, denoted as p_{ij}, is the ratio of product of η_{ij} and τ_{ij} in summation of all schedulable activity. After a sequence is obtained, Serial Schedule Generation Scheme is used to obtain an optimal schedule. Six type of heuristics are considered.

Hartmann [12] proposed a self-adapting genetic algorithm for RCPSP. The algorithm is interpreted by activity-list representation. An individual contains an activity sequence and a decoding method. A specific schedule is decoded with serial or parallel SGS from activity list λ. Two random integers q_1 and q_2 are selected for crossover. For ProGen-generated instance, self-adapting GA gives better average critical path lower bound in 1000 iterations and 5000 iterations, respectively.

A simulated annealing (SA) algorithm is discussed by Bouleimen and Lecocq to minimize project makespan [13]. A new search scheme is used to replace traditional SA search scheme. For a current solution x, a neighbour point x' is generated by shifting a random activity to a position between its earliest predecessor and latest successor. The difference $\Delta = f(x') - f(x)$ is calculated, and x' is accepted as new x if $\Delta \leq 0$ or with possibility $P = e^{(-\Delta/T)}$ if P is within acceptance range.

An exact branch-and-cut method is proposed in Zhu [17] for MRCPSP. A cut procedure is used to reduce variables and constraints after each branching and tighten the linear programming relaxations. During branching procedure, a local search is applied on the neighborhood of current solution for early-stage advantage. Some problem-specific information is used to accelerate the obtaining of LP bounds. Numerical experiments are implemented on 20- and 30-activity benchmark problems developed by Kolisch et al. and at least 502 of J30 problem set are reported with better solutions.

Valls, Ballestin and Quintanilla [19] demonstrated a Hybrid Genetic Algorithm (HGA) for deterministic RCPSP. This HGA uses an improved crossover operator to identify combine good parts of generated solutions instead of random selected part. Besides, two parents in a couple are chosen in different manner. One is selected as the fittest individual in population and the other is randomly selected. And a scheme called Double Justification is introduced to obtain a left and right active schedule. Experiment shows that HGA improves most tested algorithms in solution quality sense.

A Hybrid Scatter Search/electromagnetism Method is applied to RCPSP by Debels et al. [20] They use an improved random-key representation by eliminating time scaling and timing anomalies problem.

Jarboui et al. [21] introduce a combinatorial particle swarm optimization technique for multi-mode RCPSP. A dummy variable for each particle is created to allow transition between combinatorial status and continuous status. The velocity of a particle in next moment depends on its current velocity, distance from best location of itself and best overall location. The paper uses this method for mode optimization of an existing schedule.

Mendes et al. [22] proposed a genetic algorithm based on random key representation. A delay time for each activity is introduced in chromosome along with priority. The fitness of a solution is defined by makespan and potential for improvement.

Zheng and Wang [24] developed a Multi-agent optimization algorithm for RCPSP. Several groups of agents proceed towards optimal solution under guide of local best agents and global best agents. Solutions are improved through social behavior and right-shift based self-learning with adjustment to environment. Experiments shows that MAOA works efficiently on medium and large-scale problems.

Drezet and Billaut [28] considered a situation in labor arrangement company where resource requirement varying with time. Resource in this research is software company employee with different skill level to execute activities. Resource request may vary with time between a minimum and maximum level. Like realistic situation, employees may work only for a certain amount of time.

Mika et al. [27] considered setup time application in multi-mode RCPSP. Setup time is a pre-process for general-purpose resource to get prepared for execution of activity. It depends on the target activity, involved resource type and specific activity sequence. Furthermore, concept of location is introduced as different activity process location can affect its successors' setup time in MRCPSP.

Neumann and Schwindt [29] discussed a situation where a resource is produced by an activity and requested by another activity, which causes a storage cost during time lag. This so-called cumulative resource is maintained between a minimum level and maximum level through consumption and replenish from activities, which may cause extra holding costs or resource conflict. Instead of using activity representation, they use events to represent consumption and replenish point for cumulative resource.

A bonus-penalty policy under multi-mode RCPSP with material ordering is discussed by Zoraghi et al. [30]. In this research non-renewable resource need to be ordered in advance. Objective is to minimize total cost which consists of material order cost, holding cost, earliness bonus and tardiness penalty. Untimely non-renewable resource order or inappropriate method may result in tardiness penalty or extra inventory cost. They used a hybrid heuristic method which has an inner search stage and a multi-algorithm-based outer search stage.

In multi-mode RCPSP, two or more activities need to be operated in a fixed mode set. This mode identity constraint is discussed by Salewski et al. [31] All jobs are partitioned into finite config subsets and considered as temporal subsets. Earliest start time and latest finish time are also calculated in subset format, which greatly decrease complexity.

Erenguc et al. [32] allows the reduction of activities in project at additional resource cost. For each mode of an activity, a normal duration and a crashed duration are given. Resource constraints are expressed conceptually in this research. Actual activity duration varies between two given duration under selected mode with additional cost spent on it. Mode cost, crash cost and tardiness penalty comprise final cost which is to be minimized. Actual duration is considered as a variable. Crashing activity can be considered as a special situation of multi-mode projects.

In supply chain management, quality is considered important but hard to be quantified. Li and Womer [33] proposed a model with quality level constraint. For each activity mode, a different quality level is given. Quality level works as a constraint. A minimum reliability must be satisfied for all activity quality level summation. Objective is to minimize project duration, but total inventory cost is also discussed under general temporal constraints and variable resource capacity. A similar approach is used to maximize quality level when observing budgets and deadline in Tareghian[34].

Tiwari et al. [35] discussed quality under heterogeneous resource circumstances. In their research, an activity can be started with a resource with low level skill and completed in a rework mode. This two-step scheme can be considered as a special case of varying resource request with multi-mode activity. A new job can only be started after the rework of its predecessor is done. This rework mode usually happens in a multi-stage activity which requires consolidation after first step, like new employee training or Integrated Processing. Rework mode provides the possibility for more flexible scheduling at additional resource cost.

Demeulemeester et al. [36] demonstrated a computational result for discrete time/cost trade-off problem. Duration of activity is a non-increasing function of additional cost. With given amount of cost, a minimum project duration is found with a deterministic procedure. Their algorithm uses a backtracking technique which continuously improve project deadline. Resource constraints are not considered in their model.

Drexl et al. [37] considered cases in which activities are not allowed to be scheduled during certain periods, which is called forbidden periods. This forbidden period is related to real world time. For example, a job that must be finished without pause but is scheduled at the end of a day, which causes a suspension of ongoing process. Forbidden periods provide an approach for scheduling considering vacations, temporary shutdown and other situations that pause processing jobs.

In RCPSP, we usually assume that successor activities can be initiated as soon as their predecessor finish. In practical situation, an activity may not be started right after its predecessor. Concept of minimal time-lags is introduced as work continuity constraint by Vanhoucke [38]. Duration is substitute with time-lag which depends on two related activities and their unit level in order to match repetitive characteristic of construction projects. By using unit notation, repeating activities are integrated as one activity with same property.

In mechanical manufacturing, a material needs to be processed as soon as possible after a post-process. In that case, a successor of post-process must be started within deteriorate time. This leads to a maximal time-lag concept similar to minimal time-lag. Bartusch et al. [39] considered general time-window constraints in job shop scheduling problem. Four types of time-lags are considered: start-to-start lags (S-S), start-to-finish lags (S-F), finish-to-start lags (F-S), finish-to-finish lags (F-F) are transformed into a standardized form.

A general case of precedence relationship is release date and due date of activities. A release date is a moment only after which certain activity can be started. Similarly, a due date is a moment before which a job must be finished. Essentially a release date can be transformed into a start-to-start lag between dummy start and considered activity, while a due date can be viewed as a finish-to-start lag between designated activity and dummy source. Due to technical issues, project may take more time than expected to be complete and due date is violated. Özdamar et al. [40], Neumann et al [41] and Klein et al [42] discussed circumstances where due date can be violated by paying some penalty cost. Objective is usually to maximize net present value, but tardiness itself is sometimes considered as objective.

In industrial practice, an activity can have more than one execution mode due to resource variations. This leads to multimode resource constrained project scheduling (MRCPSP). When executed in different mode, an activity consumes different number of resources and yields different duration. Vanhoucke et al [43] generalized MRCPSP with metaheuristic approach.

An extension on activity finish time is discussed by Hartmann and Kolisch [9]. The extension forbids certain activities to finish at the same period. This is related to the case in large-scale experiments where there are multiple samples to record, or in factories where storage is not enough to hold all middle products.

Möhring, Schulz, Stork and Uetz[14] demonstrated a minimum cut method for solving project scheduling problem. They use a Lagrangian relaxation method turn original problem into a minimum cut problem.

Arkhipov et al. [24] developed a pseudo-polynomial algorithm to determine lower bound on the RCPSP problem makespan with time-dependent resource capacities. The idea is based on continuous assessment on resources capacities and workload. Start time and deadline of activities are modified repeatedly with constraints of resource under time horizon T, which is narrowed by binary search in the end of a cycle.

A Tabu search method for RCPSP with schedule-dependent setup time is proposed by Mika et al. Besides traditional renewable resources, a new type of resources called setup-required resources for preparing multi-purpose resources is introduced. The setup time consists of sequence-independent setup time, sequence-dependent setup time and schedule-dependent setup time and is related to setup-required resources locations. This separation can decrease activity complexity for large project. And satisfying result can be obtained by Tabu search metaheuristic.

2.4 Remanufacturing system

In recent decades, remanufacturing receives much attention for its potential to reduce environment pressure, pollution, and the revenue it brings to project. By remanufacture used items, project managers can reduce material cost, simplify supply chain, and schedule setup time more precisely. For project involving single item with high value, remanufacturing, instead of fabricating a new item, can considerably decrease lead time and cost needed for the project.

One of the hardest parts of remanufacturing scheduling is disassembly scheduling since recycled material and item demands are highly unstable and unpredictable. To establish connection between supply and demands, bill of material (BOM) and materials requirement planning (MRP) system are used for remanufacturing management.

Gupta and Taleb [43] introduced concept of disassembly scheduling. Reversed materials requirement planning (RMRP) is used to satisfy material requirement for components and subassemblies instead of assemblies. RMRP provides convenient access for floor material handling and inventory management, but it is hard to tell the source of demand for leaf items. In this procedure, requirement of material is set as final objective.

Taleb, Gupta and Brennan [45] present a disassembly structure graph considering part commonality. Commonality denotes that an item has more than one parent or can be yield from disassembling a common part. In their algorithm, common parts are split into separate categories stem from different parent items. Common parts in different categories, though they denote the same real-world items, are labeled with distinct number. They also show how material requirement can be met when leaf items have material commonality.

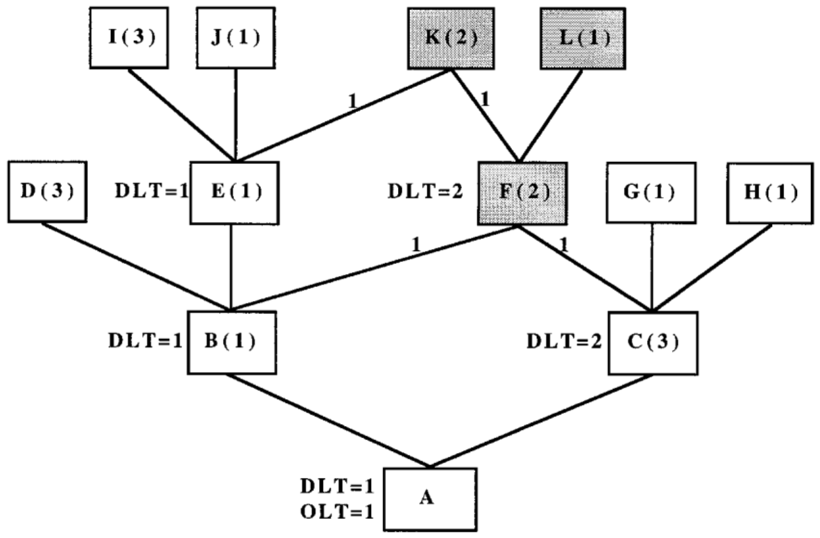


Fig 2.1 Disassembly structure with part commonality [44]

After Gupta and Taleb proposed their RMRP procedure, various extensions are introduced for remanufacturing. Lee and Xirouchakis [44] considered cost minimization as their objective. In their model, requirement is stated as leaf material. Root items are disassembled into leaf items to satisfy periodical leaf item demands, and scraped components require holding cost each period. Final cost consists of disassembly operation cost, material ordering cost, inventory holding cost and setup time cost. A mathematical model is construct and a two-stage heuristic method is used to solve the problem.

disassembling a common part. In their algorithm, common parts are split into separate categories stem from different parent items. Common parts in different categories, though they denote the same real-world items, are labeled with distinct number. They also show how material requirement can be met when leaf items have material commonality.

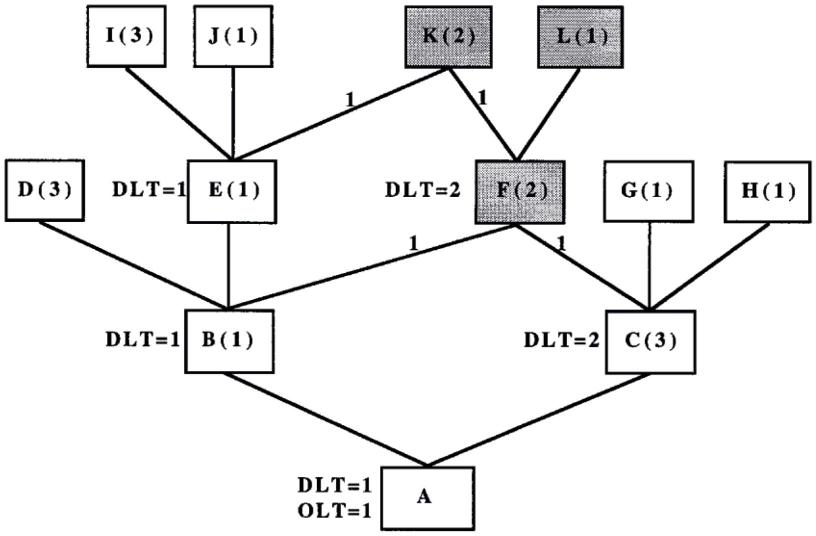


Fig 2.1 Disassembly structure with part commonality [44]

After Gupta and Taleb proposed their RMRP procedure, various extensions are introduced for remanufacturing. Lee and Xirouchakis [44] considered cost minimization as their objective. In their model, requirement is stated as leaf material. Root items are disassembled into leaf items to satisfy periodical leaf item demands, and scraped components require holding cost each period. Final cost consists of disassembly operation cost, material ordering cost, inventory holding cost and setup time cost. A mathematical model is construct and a two-stage heuristic method is used to solve the problem.

The model proposed by Gupta and Taleb is discussed in a more graphic way and not a mathematical model. An integer programming model is proposed by Lee et al. [46] to demonstrate the ordering and disassembly of end-of-life products over a planning horizon with various extensions. The model starts in a basic form considering only single product disassembly without part commonality. Objective of their integer model consists 4 parts: disassembly operation cost, setup cost, material holding cost and purchasing cost. For part commonality scenario in single product case, stock level constraint is changed such that inventory balance for each non-root item comes from multiple parent items, and demands are shown in leaf item.

Though many scheduling methods of disassembly activity has been proposed, few research in mathematical essence of disassembly scheduling was done. In Kim et al. [47], they proved that a disassembly scheduling with assembly product structure is a NP-hard problem, which means a heuristic method is needed for solving the problem. They used a Lagrangian relaxation method to obtain upper and lower bound of each node on a branch-and-bound tree. The node is taken as new incumbent solution if its next activity in sequence has better lower bound and considered as current optimal.

Inventory level is always a concern for project managers. Keep a high level of inventory increases robustness of manufacturing system but results in extra holding cost. Low inventory level saves inventory cost while brings high possibility of stock depleted. Furthermore, disassembly capacity is not unlimited in practical situation especially for large scale disassembly activity and large single items. Kim, Lee and Xirouchakis [48] considered a situation where capacity for disassembling activity is limited. In each time period, disassembly activity can only happen limited times. The objective is to minimize add-up cost of setup cost, inventory cost and disassembly cost. While a peak hour demand may still occur, it is not possible to massively disassembly items for large material requests in next period.

Capability for remanufacturing is not unlimited in most cases. In given time period, one type of parent items can be disassembled only for a certain amount of time. For example, if a large amount of leaf items is demanded at time t, it may not be feasible to disassemble parent items to acquire the leaf items needed in previous period $t-1$, given that capacity is constrained. Kim et al. [49] discussed a case in single item scenario without part commonality with capacity constraint. Their objective is to minimize the number of disassembled root item. Only limited number of items can be disassembled in each period.

In this chapter, literature concerning resource constrained project scheduling problem and remanufacturing scheduling are discussed. For RCPSP, different variants and extensions considering for temporal constraints, resource constraints and objective functions in practical situation greatly increase complexity and lead to application of different solution methodology. For remanufacturing, scenario concerning single product without part commonality is mainly discussed. In next chapter, we will present a mathematical model to demonstrate RCPSP with remanufacturing activity.

Chapter 3 Mathematical Model

3.1 Introduction

In this chapter, resource constrained project scheduling problem (RCPSP) with remanufacturing activities is studied. We give general definitions of parameters, decision variables and resource constraints. Basic assumptions will be given. Mechanism of remanufacturing process will be discussed. The model is based on Zoraghi [30] with variants of remanufacturing features. An Activity-on-Node (AoN) network will be used to present activities and their precedence. The model is developed as a multi-mode resource constrained project scheduling problem (MRCPSP). Each activity can be processed in multiple execution mode at different resource consumption rate and durations. A multi-mode project network diagram will be used for definition demonstration.

3.2 Definitions

An activity can be operated in different modes. Each mode consumes different number of renewable resources and non-renewable resource and yields different activity durations. All activities are non-preemptive, which means a project must be finished once it is started without pause and interruption.

3.2.1 Activity

An activity has a set of predecessors and a set of successors. There are four types of time lags: finish-to-start time lag, finish-to-finish time lag, start-to-start time lag and start-to-finish time lag. All four types can be transferred to finish-to-start (F-S) time lag with consideration of activity duration. A F-S time lag is the minimal time period from the complete of an activity to its successor. An activity cannot be started until its predecessor finish. Once an activity initiates, it occupies certain amount renewable resource and consumes non-renewable resource and cumulative resource. For remanufacturing activities, they produce cumulative resource which can be used later by other activities.

A dummy start and a dummy end are added to original project. They have only one mode that consumes no resource with a task duration of zero.

3.2.2 Resource

Three types of resources are considered in this study: renewable resources, non-renewable resources, and cumulative resources. A renewable resource can be used repeatably and refresh in each period, such as labor and machine. Once it is occupied by one activity, it cannot be used by other activity until current activity is finished. A non-renewable resource is a resource that has a limit of total usage, such as materials. When used by an activity, it is consumed and cannot be used anymore in project scope. Cumulative resource is a non-renewable resource. The difference between non-renewable resource and cumulative resource is that non-renewable resources are acquired from external supplier via orders and cumulative resource can only be supplied from remanufacturing activities. A non-renewable resource delivery can be obtained by placing an order that specifies material amount.

All these three resources can have different types. In our study, R_r types of renewable resources, R_n types of non-renewable resources and R_c types of cumulative resources are considered. In each period, amount of available resource is refreshed. The inventory level for non-renewable resources and cumulative resources are zero at the beginning of activities. An order needs to be placed for non-renewable resources supply from external source. A remanufacturing activity need to be done to obtain cumulative resource supplement.

Before project begins, requirement of resources by different activities and modes are known for deterministic until project complete, and the initial amounts of non-renewable resources and remanufacturing activities are zero.

3.2.3 Remanufacturing activity

Remanufacturing activities occupies renewable resources and produce cumulative resources for later used. Remanufacturing activities has no predecessors and successors. They can be operated at any time if renewable resource satisfies the condition. It is assumed that material required for remanufacturing are obtained previously and does not have a cost.

Remanufacturing activity can be considered as a predecessor for activities that required remanufactured items. Successors cannot be initiated unless remanufactured items are prepared. Since disassembly tree exist, multiple remanufacturing activities may provide cumulative resources for one manufacturing task, and one remanufacturing activity may only be start after its required item is disassembled from parent item.

In the problem, two types of remanufacture items are discussed. In the first hierarchy, all three remanufacturing activities produce same cumulative resource, but they are in precedence relation. This denotes a type of product which yields multiple recycled components that has different process time. Second one may generate 2 type of remanufactured items in remanufacturing activity 2 and 3.

In this study, disassembly structure is a single product structure without part commonality. Only one type of root item is considered at the same time, and different parent items can't be scraped into same child item.

3.2.4 Cost

The objective of this study is to minimized total cost spent on project. There are six types of cost: order cost, non-renewable resources ordering cost, non-renewable resource inventory holding cost, cumulative resource holding cost, bonus and penalty for earliness and tardiness of project.

Order cost is a fixed cost for placing an order. It is related to issuing a payment or invoice, cargo inspection, moving received goods to stock and other activity that are not related to order size. In mathematical model, a fixed amount of order cost is issued when an order for non-renewable resource is placed.

Non-renewable resource is usually related to consumable materials. Basic number of materials are needed for each single task, but increased material investment can accelerate task completion. Therefore, a trade-off between extra material purchase and bonus/penalty regarding project completion exist, and non-renewable resources need to be considered as a variable.

Inventory level is one of the most important factors for project managers. High stock level can strengthen production robustness against unpredictable material delivery delay but increase cost for inventory management. Low stock level can greatly reduce inventory holding cost while brings high risk of stockout. In our model, no backlogging is allowed.

Cumulative resources are produced by remanufacturing activities. Since remanufacturing activities are dependent from normal tasks, it is possible to finish all disassemble activity before starting a normal activity. However, early disassemble activities result in extra holding cost for recycled cumulative resource. In single item disassembly hierarchy without part commonality, recycled child items tend to be high value precision component that requires special environment to store. Remanufacturing activities, therefore, should be scheduled along with normal activities. Since non-renewable resource and cumulative resource are determined after the inspection, no opportunity cost is incurred, which means inventory cost is related to material storage.

For each project there is a due date to follow. If a project is finished before due date, a bonus specific to completion date will be paid to contractor. In contrast, a penalty with respect to finish date will be applied if project completion time does not meet due date as a compensation for not fulfilling contract.

3.3 Assumptions

This model is based on research proposed by Zoraghi [30] and Lee et al [46]. In this mathematical model, project will be demonstrated in an Activity-on-Node (AoN) network $G(V,E)$. All activities are non-preemptive, which means one activity must be kept running until it is complete and cannot be paused for reallocating resources. All activities have known and determined duration and consumption for each mode. A dummy start and a dummy end are added to original project network. Dummy start takes no resources and time to operated and is the predecessor of all activities. Conversely, dummy end requires no resource and time and is the successor of all activities.

Remanufacturing activities and normal activities are independent in precedence relations but share same renewable resources. Available number of renewable resources is given in advance. Inventory size for non-renewable and cumulative resources has a limit. For renewable resources, once used by an activity, it will stay occupied and cannot be applied on other activities until current process is finished. Non-renewable resources will be restocked to ensure no back logging happens. To get restocked, an order with specific resource number and type will be placed. A delivery lead time is needed for orders to fulfill. Cumulative resource can only be supplied received when remanufacturing activities is finished. Non-renewable resource and cumulative resource are both consumed once activity starts, and a delivered order can only be available from the second day.

Order cost for each material type is fixed and purchase cost for each unit of non-renewable resource is known and constant. Inventory cost is the same for nun-renewable resource and cumulative resource per unit per period. There is a known due date for entire project when project begins. If project is finished before due date, a bonus per date will be earned. In contrast, a penalty will be applied according to tardiness.

3.4 Mathematical model

The mathematical model will use Fig 3.1 as demonstration. Parameters, sets, decision variables and objective functions are shown below. Remanufacturing activities are related to normal activities through cumulative resource, and precedence among remanufacturing activities are determined by recycled cumulative resources. To operate a remanufacturing activity, a set-up cost is usually needed. In our model, set-up cost is considered as a part of non-renewable resource purchase cost.

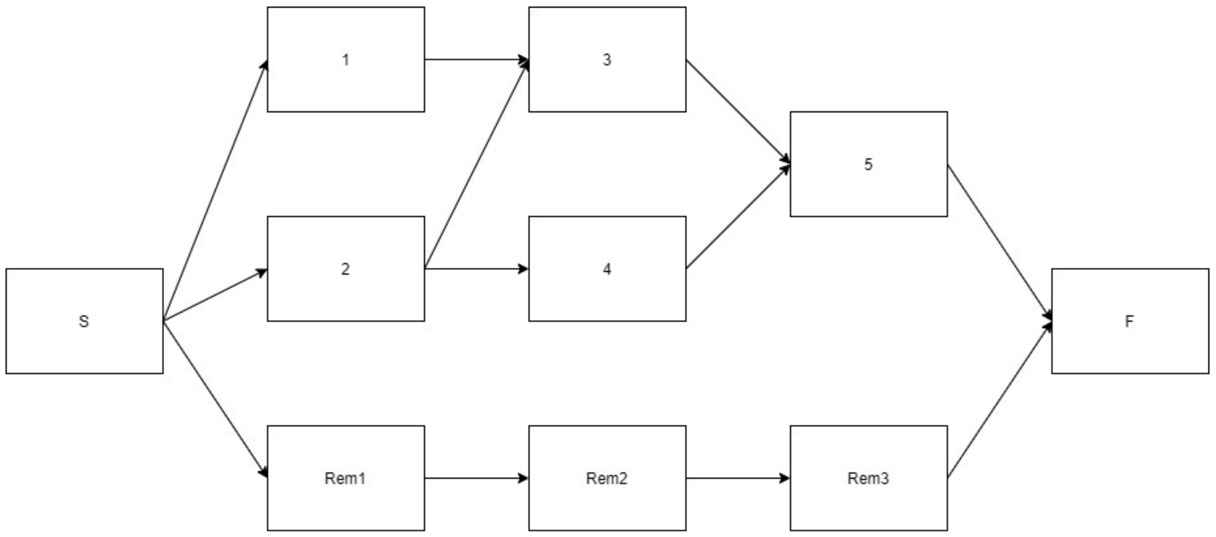


Fig 3.1 Sample project networks and information

Notations

Sets:

$TP = \{0, 1, \dots, T\}$, Set of time period

$AM = \{1, 2, \dots, M\}$, Set of normal manufacturing activities

$AR = \{1, 2, \dots, R\}$, Set of remanufacturing activities

$RS = \{1, 2, \dots, S\}$, Set of renewable resources

$RN = \{1, 2, \dots, N\}$, Set of non-renewable resources

$RC = \{1, 2, \dots, C\}$, Set of cumulative resources

M_i, Set of modes of activity i, $i \epsilon\ AN \cup AR$

$E = \{(i, j) |\ i, j \in AN \cup AR\}$, Set of precedence relations,

Parameters

O_n, Order cost for nonrenewable resource n each time per order

P_n, Price for per unit of non-renewable resource n

H_n, Holding cost for per unit of non-renewable resource n per time period

L_c, Holding cost for per unit of cumulative resource c per time period

P, Penalty applied for late finish after due date per day

B, Bonus applied for early finish before due date per day

DD, Due date of project

NL, A large number

R_s, Max capacity of renewable resources s that can be used in each time period

D_n, Delivery lead time of non-renewable resource n after placing order

d_{im}, Duration of activity i when operate in mode m

w_{ims}, Consumption of renewable resource s when activity i operates in mode m

t_{imn}, Consumption of non-renewable resource n when activity i operates in mode m

u_{imc}, Consumption of cumulative resource c when activity i operates in mode m

v_{imc}, Produced cumulative resource c when remanufacturing activity i operates in mode m

Decision Variables

$$s_{imt} = \begin{cases} 1, \text{if activity } i \text{ starts at time } t \text{ in mode } m \\ 0, \text{otherwise} \end{cases}$$

$$r_{emt} = \begin{cases} 1, \text{if remanufacturing activity } e \text{ starts at time } t \text{ in mode } m \\ 0, \text{otherwise} \end{cases}$$

$$o_{nt} = \begin{cases} 1, \text{nonrenewable resource } n \text{ is ordered at time } t \\ 0, \text{otherwise} \end{cases}$$

i_{nt}, inventory level of nonrenewable resource n at time t

v_{ct}, inventory level of cumulative resource c at time t

m_{nt}, the amount of nonrenewable resource n ordered at time t

Objective Function

$$\begin{aligned} minimize\ Z = & \sum_{n=1}^{N}\sum_{t=1}^{T} O_n \times o_{nt} + \sum_{n=1}^{N}\sum_{t=1}^{T} P_n \times m_{nt} + \sum_{n=1}^{N}\sum_{t=1}^{T} H_n \times i_{nt} + \sum_{c=1}^{C}\sum_{t=1}^{T} L_c \times v_{ct} \\ & + \sum_{t=DD+1}^{T} P \times (t - DD) \times s_{Mmt} \\ & + \sum_{t=EF}^{DD-1} B \times (t - DD) \times s_{Mmt}, \end{aligned} \tag{1}$$

Constraints

$$\sum_{m=1}^{M_i}\sum_{t=1}^{T} s_{imt} \times (t + d_{im}) \leq \sum_{m=1}^{M_j}\sum_{t=1}^{T} s_{jmt} \times t\,, \forall (i,j) \in E, \tag{2}$$

$$\sum_{i=1}^{M}\sum_{m=1}^{M_i}\sum_{w=Max\{t-d_{im},ES_i\}}^{Min\{t,LS_i\}} s_{imw} \times w_{ims} + \sum_{e=1}^{R}\sum_{m=1}^{M_e}\sum_{w=Max\{t-d_{em},ES_e\}}^{Min\{t,LS_e\}} r_{emw} \times w_{ems} \leq R_s,$$

$$\forall s \in RS, \forall t \in TP, \tag{3}$$

$$i_{nt} = i_{n,t-1} + m_{n,t-D_n} - \sum_{i=1}^{M}\sum_{m=1}^{M_i} s_{imt} \times t_{imn}\,, \forall n \in RN, \forall t \in TP\,, \tag{4}$$

$$\sum_{i=1}^{M}\sum_{m=1}^{M_i} s_{imt} \times t_{imn} \leq i_{n,t-1}, \forall n \in RN, \forall t \in TP, \tag{5}$$

$$v_{ct} = v_{c,t-1} + \sum_{e=1}^{R}\sum_{m=1}^{M_e} r_{em,t-d_{em}} \times v_{emc} - \sum_{i=1}^{M}\sum_{m=1}^{M_i} s_{imt} \times u_{imc},$$

$$\forall c \in RC, \forall t \in TP, \tag{6}$$

$$\sum_{i=1}^{M}\sum_{m=1}^{M_i} s_{imt} \times u_{imc} \leq v_{c,t-1}, \forall c \in RC, \forall t \in TP, \tag{7}$$

$$\sum_{m=1}^{M_i}\sum_{t=1}^{T} s_{imt} = 1\,, \forall i \in AM, \tag{8}$$

$$\sum_{m=1}^{M_i}\sum_{t=1}^{T} r_{emt} = 1\,, \forall e \in AR, \tag{9}$$

$$m_{nt} \leq o_{nt} \times NL, \forall n \in RN, \forall t \in TP, \tag{10}$$

$$i_{n0} = 0, \forall n \in RN, \tag{11}$$

$$i_{nt} \geq 0, \forall n \in RN, \forall t \in TP, \tag{12}$$

$$v_{c0} = 0, \forall c \in RC, \tag{13}$$

$$v_{ct} \geq 0, \forall c \in RC, \forall t \in TP, \tag{14}$$

$$m_{nt} \in Z, \forall n \in RN, \forall t \in TP, \tag{15}$$

Objective is to minimize total cost spent to finish projects. Objective function consists of 6 terms: order cost for non-renewable material; purchase cost for non-renewable material; inventory holding cost for non-renewable resource; inventory holding cost for cumulative resources; penalty for project late finish and bonus for project early finish.

13 constraints are used in model. Precedence relations are guaranteed by inequality (2). Inequality (3) makes sure that renewable resource usage is respect in each period. Equation (4) describes that current non-renewable inventory level depends on last period inventory, delivered orders and total consumption in this period. Inequality (5) ensures that activities can only use non-renewable resource from previous day. Cumulative resource inventory is calculated by Equation (6), as cumulative resource will be restocked when remanufacturing activity is completed. Inequality (7) ensures that an activity cannot be scheduled unless there is enough cumulative resource in previous day. Equation (8) makes sure all normal activities are processed once and in only one mode, and remanufacturing activities is constrained by cumulative resource non-negative inventory cost. Equation (9) implies that one remanufacturing activity can only be executed once and only once. Inequality (10) enforces a non-renewable resource delivery is only considered if an order is placed. Equation (11) and (12) show that non-renewable resource inventory level is zero when project starts and must be non-negative integer. Equation (13) and (14) give zero initial inventory and non-negative integer constraint to cumulative resource. Equation (15) enforce material purchase amount to be non-negative integer.

A mathematical model is presented in this chapter. Relative terms, definitions and assumptions are demonstrated. Remanufacturing activities is considered as an extension to Zoraghi's model. Recycled product is considered as cumulative resource, which is consumed and restocked when activity starts.

In next chapter, a three-phase heuristic algorithm will be proposed to solve this mathematical model. 18 small size instances generated by RanGen1 will be used for demonstration.

Chapter 4 Solution, Method and Testing Examples

It is known that resource constrained project scheduling problem is a NP-complete problem. For practical project planning, project managers usually use commercial scheduling and optimization software, which takes long to solve a single case.

Therefore, a three-phase heuristic method is proposed for solving RCPSP with remanufacturing activities. In this section, 20 small instances generated by RanGen1 are used for evaluation.

To demonstrate the algorithm, a sample project with 3 possible operation modes for each activity is used. Details are shown below.

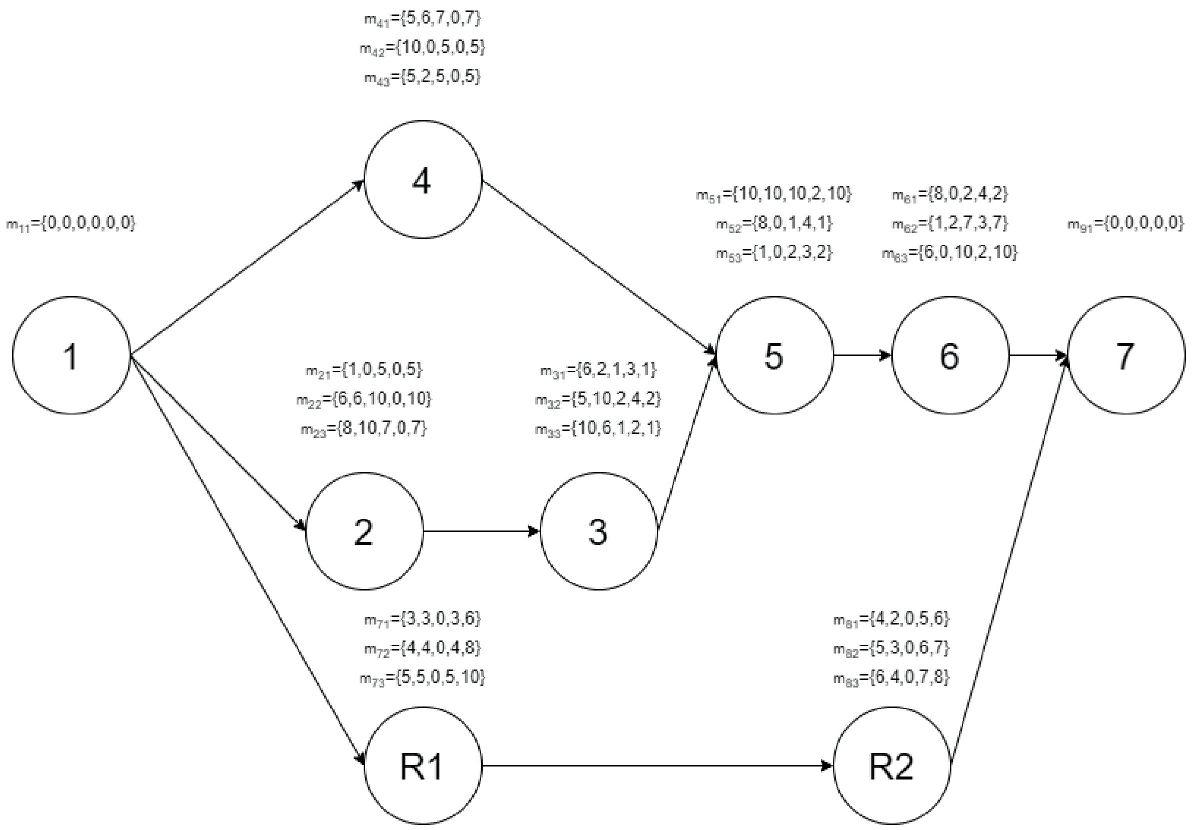


Fig 4.1 Sample project for algorithm demonstration

4.1 Three-stage heuristic method

As it was mentioned, a resource constrained project scheduling problem is a NP-complete problem. Multi-mode RCPSP with remanufacturing activities, therefore, is more complex. Computational cost for generating an optimal solution grows exponentially with project scale. Exact method like Branch-and-Bound may take days to solve a small-scale project. For large-scale project, classic methods often fail to find a solution.

Heuristic methods are approach that ensures a feasible solution that is not guaranteed optimal within limited times or attempts, which solve a problem with acceptable time cost. One way to generate a heuristic method is to break the problem into several sub-problems. Each sub-problem has an objective function and constrains by several constraints from original problems. The output from previous sub-problem is taken as the input for next sub-problem.

To solve this multi-mode resource constrained project scheduling problem with remanufacturing activities, we break the problem into three subproblem and develop a three-phase heuristic method. Each subproblem uses only parts of project data to save computing time.

Objective in first phase is determine feasible mode set considering cumulative resource. Since remanufacturing activities can be implemented in different modes with different cumulative resource yields, certain modes may not be chosen together, or resource conflict may happen for normal activities. Mode sets for remanufacturing activities gives additional resource constraint on normal activities. Second, an activity schedule based on mode set determined from phase one is generated, which minimize the sum of penalty, bonus and cumulative resource inventory cost, or to minimize the project finish time within all feasible mode space. In phase three, with the schedule obtained from previous phase, we make arrangement on non-renewable material order time and order number.

This objective is achieved by determining proper remanufacturing activity mode and time lag with related manufacturing activities. We use natural date variable as the coding method, which allows easier scheduling for resource ordering. With determined remanufacturing mode and time lag, a virtual precedence between remanufacturing activities and manufacturing activities can be calculated and project network is simplified. A partial schedule for remanufacturing activities and their virtual successor manufacturing activities can be acquired. In second phase, activity mode is decided with minimal product of resource demand and per unit cost. In the third phase, a schedule is generated with serial scheduling.

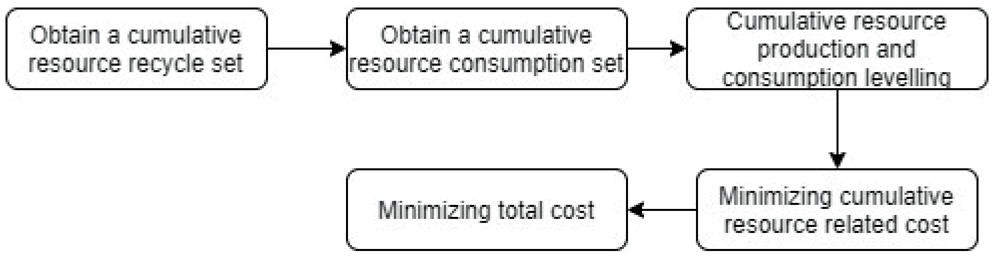

Obtain a cumulative resource recycle set
Obtain a cumulative resource consumption set
Cumulative resource production and consumption levelling
Minimizing cumulative resource related cost
Minimizing total cost

Fig 4.2 Flowchart of algorithm

4.1.1 Stage One

First phase is to obtain a substantial precedence set which ensure the cost spent on cumulative resource is minimized. As stated earlier, cumulative resource recycled by remanufacturing activities is one of the non-deterministic factors for overall inventory cost. By matching remanufacturing activities and normal activities by specific cumulative resource, a feasible mode set can be determined if remanufacturing mode is known, and search space for remanufacturing activity schedule can be narrowed.

Cost directly related to remanufacturing activity consists of one part: inventory cost for cumulative resource before they are used. It is possible that not all remanufacturing activities are completed, as long as cumulative resource inventory does not deplete during project makespan. Consequently, the objective is to minimize unused cumulative resources at the end.

In original problem, s_{imt} and r_{emt} denote the initiation time t of normal activity i and remanufacturing activity e in mode m, respectively. In the first phase, activity start time is not considered. Only activity mode is considered.

$$minimize\ Z = \sum_{c}^{RC} \left(\sum_{e=1}^{R} \sum_{m=1}^{M_e} r_{em} * v_{emc} - \sum_{i=1}^{M} \sum_{m=1}^{M_i} s_{im} * u_{imc} \right) \tag{15}$$

In this model, r_{em} and s_{im} denotes operate mode for remanufacturing activity e and normal activity i, which can be obtained from equation

$$r_{em} = \sum_{t}^{T} r_{emt}, \forall e \in AR, \forall m \in M_e \tag{16}$$

$$s_{im} = \sum_{t}^{T} s_{imt}, \forall i \in AM, \forall m \in M_i \tag{17}$$

And constraint (6), (7), (8), (12) and (13).

The objective function (15) aims to find a combination of two types of activity such that cumulative resource inventory level is minimized. Since cumulative resource inventory must be non-negative, remanufacturing activity are forced to initiate before normal activities start. Constraint (16) and (17) indicate that activity initiate time is not considered.

In sample project, cumulative resource that can be produced varies from 8 units to 12 units, depending on which mode is chosen, while total usage varies from 6 units to 12 units. If remanufacturing 1 (R1) operates in mode 1 (M1) while remanufacturing activity 2 (R2) in mode 2 (M2), 9 units of cumulative resource is produced, and normal activity cannot choose a mode set which consumes more than 9 units of cumulative resource, and two sets of precedence are set: whether A3 operates in M2. Besides, R2 is forced to become a predecessor of activity 5, since activity 3 and 5 will consume a minimal amount of 4 cumulative resource. Furthermore, if R1 process in mode 1, activity 1 cannot be operated in mode 2, which consumes 4 cumulative resources, before R2 finish. It is known that solution space decreases as precedence constraints increase. Given that R1 and R2 mode are both set, additional successors will be added for R2 to ensure non-negative inventory constraint.

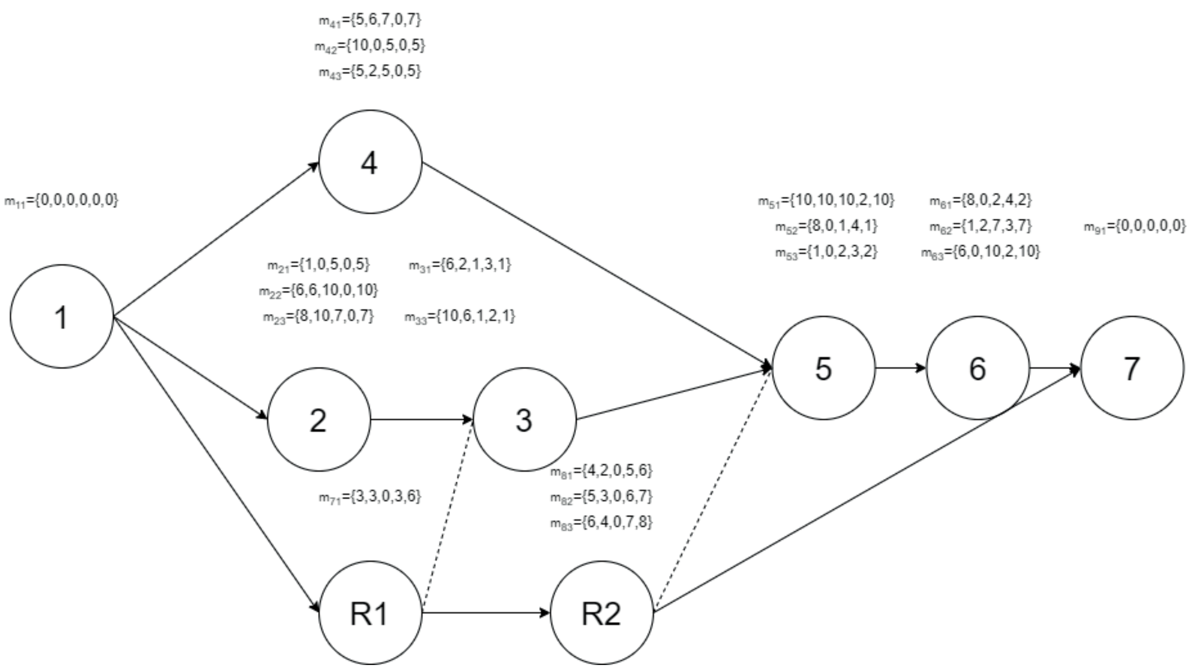


Fig 4.3 Sample project with R1 set to M1, A3 not set to M2

Table 4.1 Precedence and forbidden set for R1M1 and A3M1

R1 mode	R2	A3 mode	Forbidden set
3	5	A3M1	(A5, A6M1), (A5M2, A6), (A5M3, A6M2)
3	6	A3M1	(A5M2, A6M1), (A5M3, A6M1), (A5M2, A6M2)
3	7	A3M1	(A5M2, A6M1)

Table 4.2 Precedence and forbidden set for R1M1 and A3M3

R1 mode	R2	A3 mode	Forbidden set
3	5	A3M3	(A5M2, A6M1), (A5M2, A6M2), (A5M3, A6M1),
3	6	A3M3	(A5M2, A6M1)
3	7	A3M3	None

When normal activity 3 is set to mode 2, A3 becomes a successor of R2, and tighter forbidden set appears, since R1M1 can only produce 3 unit of cumulative resource which is not sufficient until R2 finish. As it is known, a tight forbidden set implies more precedence pairs. Search space for solution is further limited, as it can be seen in Fig 4.3 and Table 4.3.

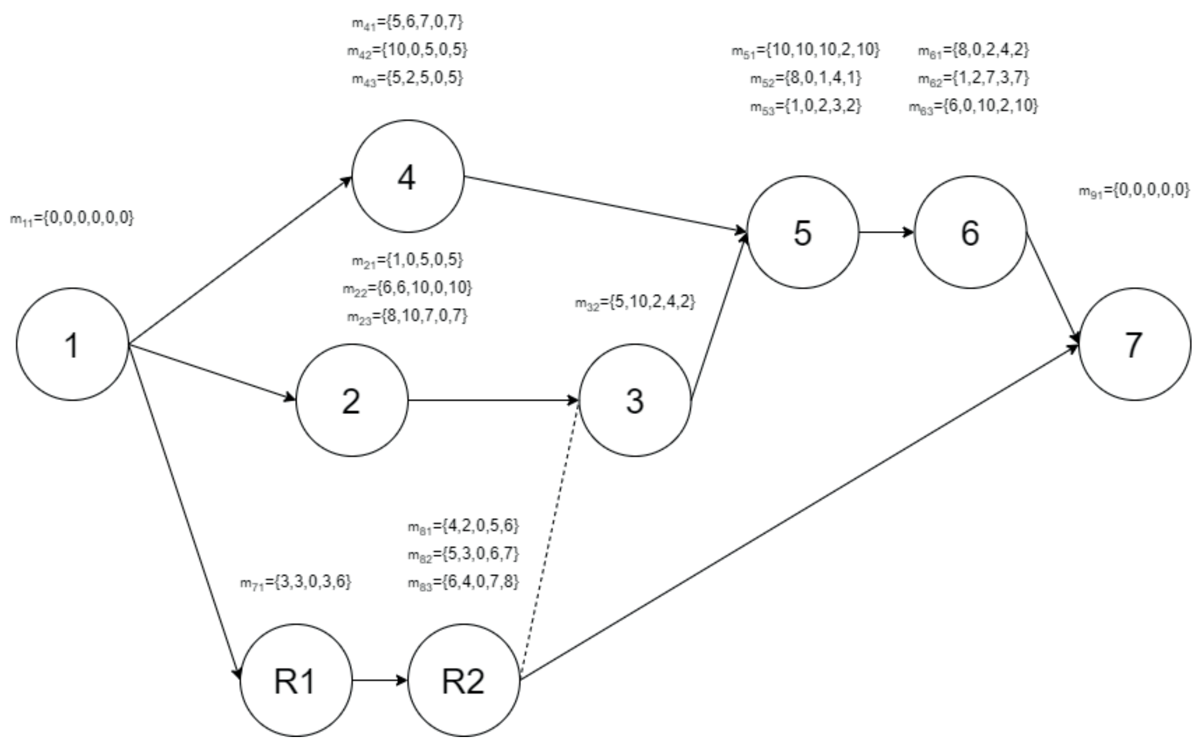


Fig 4.4 Sample project with R1 set to M1, A3 set to M2

Table 4.3 Precedence and forbidden set for R1M1 and A3M2

R1 mode	R2	A3 mode	Forbidden set
3	5	A3M2	All combination besides (A5M1, A6M3)
3	6	A3M2	(A5, A6M1), (A5M2, A6), (A5M3, A6M2)
3	7	A3M2	(A5M2, A6M1), (A5M3, A6M1), (A5M2, A6M2)

To easily append additional precedence set, a two-step algorithm is used.

Step 1

The start phase focus on total amount of cumulative resource. As it is mentioned, availability for cumulative resource is an integer between 8 and 12 unit, depends on remanufacturing activity operation mode. For each integer we can obtain a resource infeasible mode set in which the combination requires more resource than total production from all remanufacturing activity.

Cumulative resource production and consumption table us shown below.

Table 4.4 Cumulative resource consumption distribution for manufacturing sets

Resource consumption	6	7	8	9	10	11	12
Number of mode set	1	3	6	7	6	3	1

Table 4.5 Cumulative resource production distribution for manufacturing sets

Resource Production	8	9	10	11	12
Feasible mode set	10	17	23	26	27

Table 4.6 Cumulative resource production distribution for remanufacturing sets

Resource Production	8	9	10	11	12
Remanu activity set	1	2	3	2	1

Since RanGen1 can only produce network with renewable resource, here we use standardized cumulative production and usage data. If remanufacturing activity operates in low output mode, it consumes less renewable resource while producing less cumulative resource, and vice versa. In instance problem, we have a total of 9 mode possible mode set for remanufacturing activities and 27 mode set for cumulative-resource-consuming activities. These mode sets can be obtained in advance.

Step 2

The mode sets we obtained in step 1 do not specify actual activity mode and virtual precedence. In step 2 an enumeration is applied for each remanufacturing mode set to get its corresponding manufacturing mode sets.

In example problem, we start from 8 unit of resource production. Only R1M1 and R2M1 can produce a total of 8 unit of cumulative resource with R1 produce 3 unit and R2 produce 5 unit. Therefore, R1 must be complete one day earlier before the initiation of A3, and R2 becomes predecessor of A5 and A6 for any mode. The number of mode combination is finite and can be exhausted prior to calculation as a preprocess.

4.1.2 Stage Two

From first phase, multiple combination of activity modes related to remanufacturing activity is obtained, and corresponding forbidden set are determined. Original problem us therefore transferred to a traditional multi-mode resource constrained scheduling problem. In the second phase the objective is to minimize total project makespan, which means to minimize the sum of penalty and bonus, as non-renewable material resource order does not affect project finish time once the schedule is generated. Besides, cumulative resource needs to be considered in this phase, as the inventory cost for cumulative resource is determined with generated schedule.

$$minimize\ Z = \sum_{c=1}^{C}\sum_{t=1}^{T} L_c v_{ct} + \sum_{t=DD+1}^{T} P(t-DD)s_{Mmt} + \sum_{t=EF}^{DD-1} B(t-DD)s_{Mmt} \qquad (18)$$

Subject to Constraint (2)(3)(6)(7)(8)(9)(13)(14).

In this phase, non-renewable resource is not considered, as non-renewable resource consumption is positively correlated to project duration. As it can be seen, project yields zero penalty and non-zero bonus if finished before due date. On contrast, penalty will apply, and zero bonus will be received if project finish after due date. Remanufacturing activities, on the other hand, should be scheduled properly. If recycled cumulative resource stays in inventory too long, those components yield high inventory cost. This term forces remanufacturing activities to be scheduled only when resource is needed by another task.

Objective function (18) in this phase contains no information about non-renewable resource order, purchase, and inventory. To decrease calculating time for model, constraints concerning material ordering are also excluded without affecting final result.

Constraint (2) ensures precedence relations between normal activities and remanufacturing activities are followed. Constraint (3) requires that renewable resource capacity must be respected. As mentioned, this problem is a NP-hard problem, which means solution space grows exponentially as time space increase. We use time earliest start and latest start of each activity to formulate a time window that contains all possible start time for each activity. In constraint (6), cumulative resource stock is calculated. In practical situation, recycled resource requires additional time before they can be put into use. Cumulative resource restock happens after remanufacturing activity completes, and inequality (7) shows this restock cannot be used until next time point. Equality (8) and (9) gives guarantees that each task must be done once and only once. Constraint (13) and (14) ensure non-negative value and initial storage for cumulative resource.

4.1.3 Stage Three

From previous stage, a schedule that minimize cumulative resource inventory cost, project penalty and project bonus is obtained. In this schedule, the start time and execute mode for each activity are determined.

$$minimize\ Z = \sum_{n=1}^{N}\sum_{t=1}^{T} O_n o_{nt} + \sum_{n=1}^{N}\sum_{t=1}^{T} P_n m_{nt} + \sum_{n=1}^{N}\sum_{t=1}^{T} H_n i_{nt} \qquad (19)$$

Subject to constraint (4)(5)(10)(11)(12)(15).

In objective function (19), non-renewable material order place time and purchase quantity are to be decided. In this function, o_{nt} and m_{nt} are independent, while i_{nt} follows these two variables, and this is why we put non-renewable resource inventory term in phase three.

Equality (4) calculates inventory level of non-renewable resource from previous inventory, material received and consumption each day. Inequality (5) denotes that an activity which consumes more non-renewable material than previous day's inventory cannot be scheduled. As cumulative resource cost is calculated in stage two, no remanufacturing part is involved in phase three. Constraint (10) enforces an order to be placed if any amount of non-renewable resource is purchased, while Constraint (15) ensures that amount of purchased material is a non-negative integer. Constraint (11) and (12) indicates that inventory level is 0 when project starts and must be non-negative in project makespan.

4.2 Testing Examples

In this research, we use 18 small instance contains 5 manufacturing activities and 2 remanufacturing activities generated by RanGen1 network generator to evaluate the solution method.

RanGen1 is a project network generator proposed by Demeulemeester et al. [53]. With proper parameter setting, a large number of projects with renewable resource can be generated.

RanGen1 use 5 parameters to generate networks: activity number, order strength, number of resource types, resource factor and resource constrainedness. Activity number is a positive constant that does not include dummy start and dummy ends. It is positive related to CPU-calculation time, means that the larger project is, the more time it requires to generate a feasible schedule.

The definition of Order Strength (OS) is

$$OS = \frac{2|E|}{n(n-1)}$$

, where $|E|$ is the number of precedence and $\frac{n(n-1)}{2}$ is the maximum possible amount of precedence in this project. OS weighs the significance of project complexity. A larger OS value means more precedence relationship between activities in concerned project, while much precedence can be invalidated by preprocess.

Resource factor (RF) determines the density of resource usage matrix by individual activity. It is defined by Copper [54] as follows:

$$RF = \frac{1}{nS}\sum_{i=1}^{n}\sum_{s=1}^{S}\begin{cases}1, if\ r_{is} => 0\\ 0, otherwise\end{cases}$$

Where S denotes number of renewable resources, n denotes number of manufacturing resources and r_{is} denotes amount of renewable resource s during process. RF is a number between 0 and 1. If RF is close to 1, activities tend to use more resource type. Conversely, if RF is close to 0, activities tend to use only 1 resource or do not use any renewable resource at all.

Resource constrainedness (RC) is introduced by Patterson [55] as follows:

$$RC_k = \frac{\bar{r}_k}{a_k}$$

where $\bar{r}_k$ denotes average amount of renewable resource k required by all activities and a_k denotes amount of renewable resource k during whole project. RC is implemented as a measure for average resource tightness. Generally, if RC_k is close to 0, more activities use resource k can be carried out at the same time. RC_k is 1 means all activities will use resource k, and activities in this project can only be processed one by one.

The network generated by RanGen1 is saved in Patterson format. In a Patterson format file, renewable resource type number, available number of resources. resource usage for each task, duration and successors are given. The time window for activities indicating earliest start (ES) and latest start (LS) is calculated based on each individual project.

In Demeulemeester et al. [53], RanGen1 is not capable of directly generating multi-mode project instance. To introduce multi-mode project, two tradeoff functions are given: time/cost and time/resource tradeoff function and resource/resource tradeoff function. In this paper the resource/resource tradeoff function with modified work content weight is use for generating the second and third activity mode. To generate a resource usage set, we assign a work content W_i to each activity i, and a work content weight w_{is} to each activity i and renewable resource type s, for which $W_i = \sum_{s=1}^{S} w_{ik} r_{ims}$. To keep W_i and w_{is} constant, we randomly increase or decrease a resource demand, and do the opposite on another resource. Non-renewable resource consumption is considered the same as activity duration.

Generated instance consists of 5 manufacturing activities, a dummy start and dummy end. Besides, a small project consists of 2 remanufacturing activities is added parallelly to original project. The remanufacture part does not have precedence relationship with normal part. Instead, the connection is established through cumulative resources. Cumulative resource produced in each mode is related to overall Resource Strength. Since RanGen1 is only capable of generating renewable resource usage, non-renewable resource is linear to activity duration and cumulative resource total usage is between lowest and highest possible recycle amount from remanufacturing activity with a random distribution on three activity.

4.3 Working Instance

In this section we use Instance 2 for demonstration.

There are two renewable resources, one non-renewable resource and one cumulative resource in this instance. Normal manufacturing activities are labelled with number 1 to 7, while two remanufacturing activities are labelled as R1 and R2. For normal activities, total renewable resource usage and activity duration are of negative correlation. For each time period a normal activity occupies, a unit of non-renewable resource is consumed. Renewable resource has a capacity of 10 units for each type, which will be occupied by an initiated activity, and released after the activity finish. For non-renewable resource, an order with specified amount of resource needs to be placed, and the resource will be available the second day they arrive. Cumulative renewable resource consumption is preset for three normal activities with possible choice of two, three and four units, which may result in a combination of total usage from 6 to

12 units.

As it is mentioned, we assume that remanufacturing activities do not consume non-renewable resource, and the cumulative resource consumption indicates the capability of production in each mode. Renewable resource consumption and cumulative resource production are positively correlated to duration in each mode.

From Fig 4.4 we notice that all A4, A5 and A6 have a predecessor R1, as they cannot be scheduled without cumulative resource. Furthermore, R2 becomes the predecessor of A5 and A6 or A4 and A6, depending on whether A4 or A6 is scheduled first.

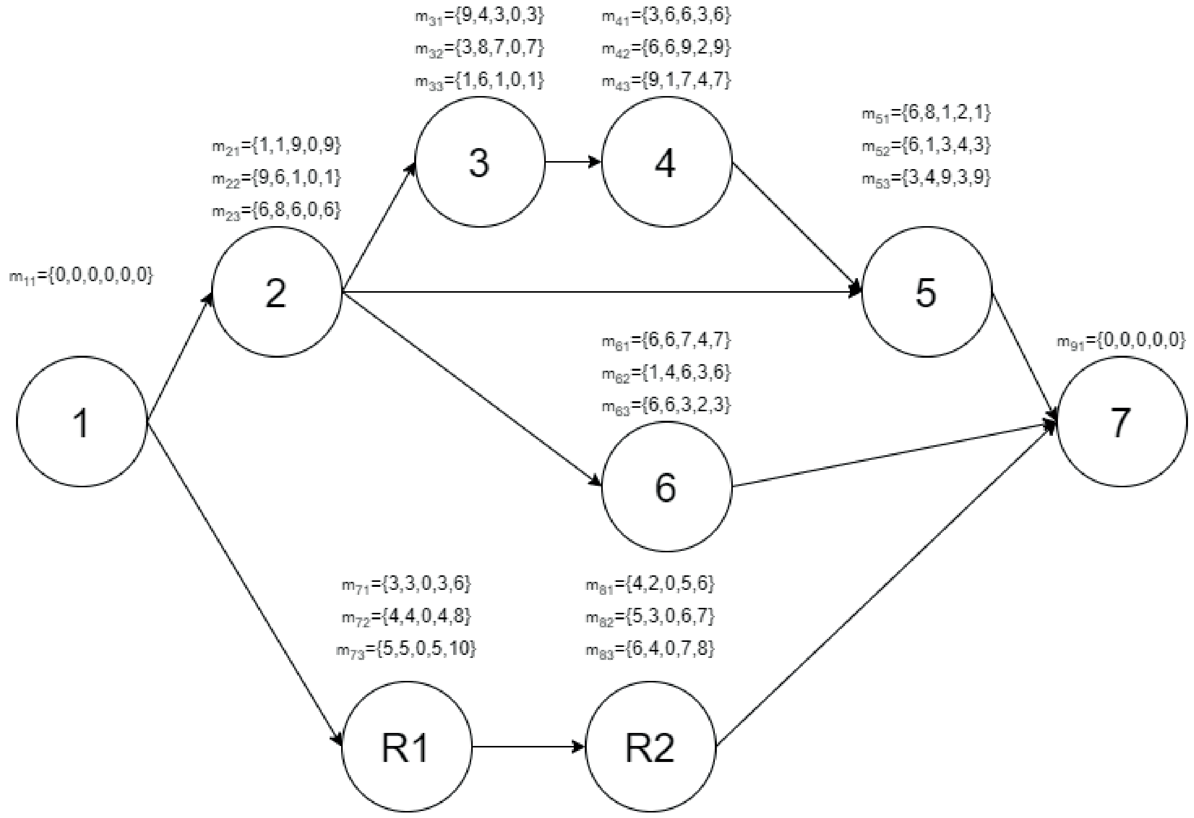


Fig 4.5 Information for Instance 2

The optimal schedule and heuristic schedule are shown in Table 4.5.

Table 4.7 Schedule for optimal solution and heuristic solution

Activity	Mode-N	Start Time-N	Mode-H	Start Time-H
6	0	24	0	23
5	1	18	1	17
4	0	17	0	16
3	0	10	0	10
2	2	3	2	8
1	1	2	1	2
0	0	1	0	1
Remanufacture 1	0	10	0	9
Remanufacture 0	0	3	0	3

Heuristic method yields a schedule close to optimal method, while postpone activity 2 for 5 days and order non-renewable resource for activity 2 and 3 separately, which makes a trade-off between non-renewable resource inventory cost and order cost.

The average renewable resource usage is shown in figures and tables as below.

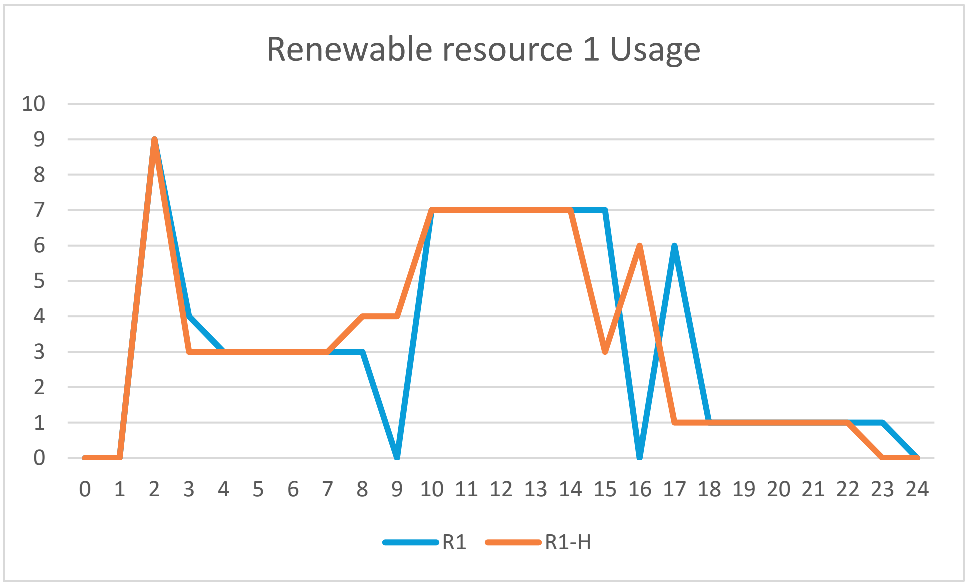


Fig 4.6 Renewable resource 1 usage for Instance 2

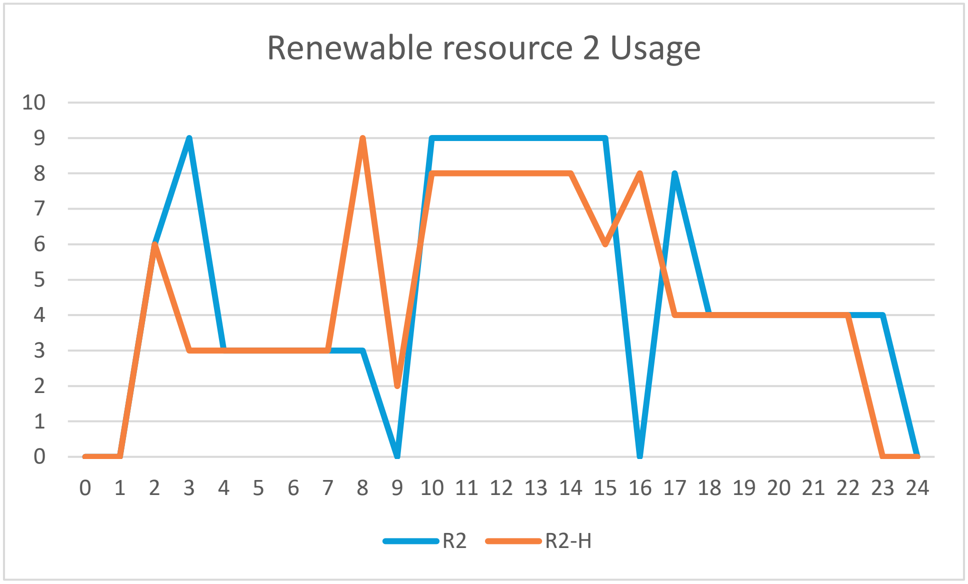


Fig 4.7 Renewable resource 2 usage for Instance 2

Table 4.8 Average renewable resource usage for Instance 2

	Normal	Heuristic
R1	0.357	0.373
R2	0.504	0.500

CPLEX solver considers all possible sequences of activities and mode combination, while heuristic method takes three steps to render a feasible schedule. In first stage, all possible cumulative resource production levels and consumption levels are enumerated in Excel. Only cumulative resource combination with same input and output level will be matched, which reduce cumulative resource inventory cost.

For each project, 10 units of renewable resource 1 and renewable resource 2, respectively, are available and refresh daily. The average renewable usage denotes the ratio of occupied renewable resource to total availability, which can be used to monitor the efficiency of resource allocation. In this instance, the heuristic method proposes a schedule with slightly higher resource 1 usage while maintains a shorter schedule. Non-renewable resource and cumulative resource can only be used one day after they are prepared, which result in the resource usage of 0 in CPLEX solver.

In Instance 2, mode 0 is selected for both remanufacturing activity 1 and 2, providing 8 units of cumulative resource in total. Consequently, activity 3, 4 and 5 will consume 8 units of remanufactured resource in total, which has 6 possible mode set. In step 2, cumulative resource inventory and project bonus and penalty is minimized, and a schedule of all activity is provided. Since additional precedence constraints are created, the complexity of sequencing is reduced, and a schedule with minimized remanufactured resource inventory cost and bonus can be quickly obtained. Since cumulative resource production is determined once the remanufacturing activity mode is considered, it must be minimized in step 2. In step 3, based on generated schedule and resource requirement, the schedule for non-renewable resource order is calculated to ensure minimized material purchase cost, inventory cost and order cost. In this model, an order for non-renewable resource takes one day to fulfill and one day to get the arrived material prepared, which means an order takes at least two day to function and generates inventory cost for at least one day. For optimal solution, 4 orders are placed at time 0, time 14 and time 15, which incurs a non-renewable resource inventory cost of 64 and an order cost of 60. The heuristic method placed 5 orders in total with the order cost of 75 in total and 60 units of non-renewable resource inventory cost.

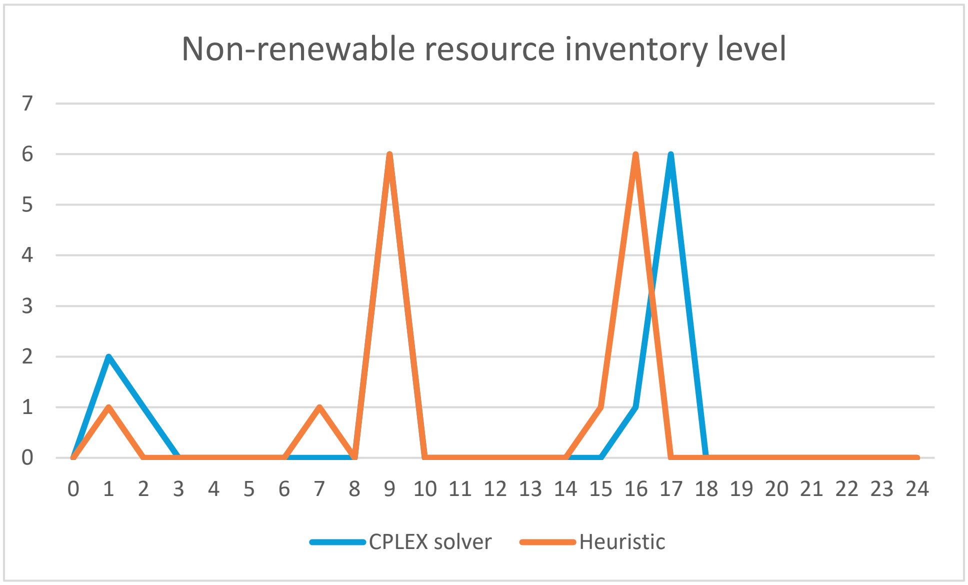


Fig 4.8 Non-renewable resource inventory level for Instance 2

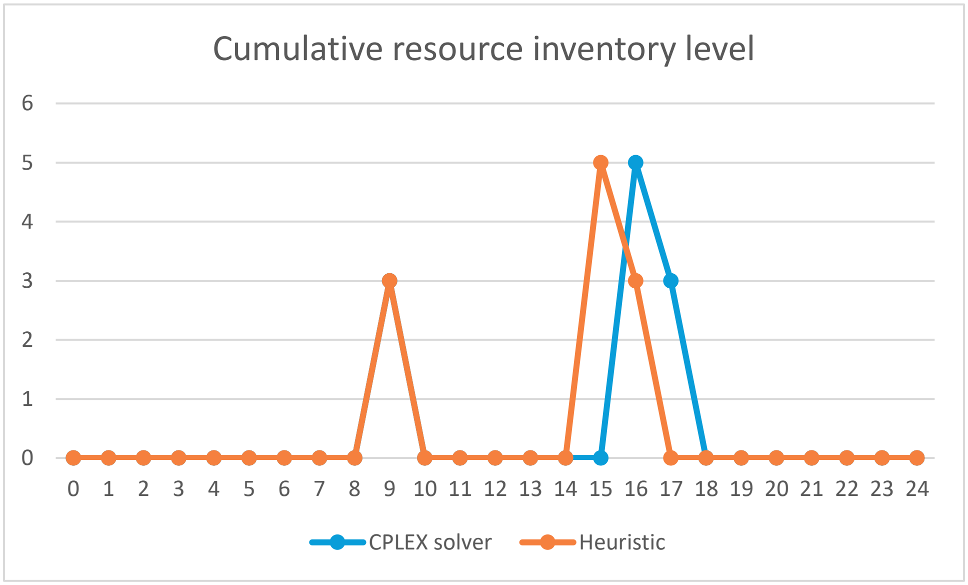


Fig 4.9 Cumulative resource inventory level for Instance 2

Result for Instance 2 from optimal solution and heuristic solution is shown in table 4.6. From the result, there is a 1% gap between two solutions. However, it takes more than 2 hours for CPLEX solver to obtain optimal solution, while the heuristic method takes only 7 seconds to get a suboptimal solution.

Table 4.9 Cost by category for Instance 2

	Bonus	CRInventory	MaterialP	NRInventory	OrderCost	Penalty	ObjectValue
Optimal	-28	44	225	64	60	28	421
Heurictic	-21	44	225	60	75	21	425

4.4 Experiment result

The instances are tested in IBM ILOG CPLEX 12.8 environment. Original problems are not able to be solved within 2 hours with embedded CPLEX solver, and known optimal solution is obtained by using embedded solver for a 2-hour computation and the results are taken as known optimal. Phase 1 of cumulative resource leveling is obtained by enumeration, while Phase 2 and Phase3 are coded in ILOG CPLEX.

In proposed model, objective function is to minimize total project cost. Since there is no research considering remanufacturing activity situation in MRCPSP, the instances are acquired from RanGen1 with 2 renewable resource with 1 non-renewable resource and 1 cumulative resource added. Therefore, the known optimal solution is obtained from CPLEX embedded solver.

RanGen1 requires following network-related parameters to generate project network: number of activities, Order-strength value, number of resources, resource factor value and resource constrainedness value. The order strength denotes how well the project network is sequenced. A bigger order strength means more precedence relations between activities. Renewable resource has a default capacity of 10 for each resource type. Resource factor denotes how intense the resource is used. This value is set to 0.9 or 1, which means there is at most one activity requires only one renewable resource type, while all other activities requires both two renewable resource to initiate. Resource constrainedness is the measure for average resource usage. A big constrainedness value means that an activity tends to use more renewable resource, which makes it less possible to operate two activities parallelly due to resource constraint. This value is set to be 0.5 or 0.75. Table 4.4 gives an overall description of all parameters for generating instances used in this paper.

Table 4.10 Parameters used in RanGen1 for generating instances

Parameter for generating instance	
Non-dummy activity number	5
Remanufacturing activity number	2
Order Strength	0.5/0.75
Renewable/Non-renewable/Cumulative resource types	2/1/1
Renewable resource capacity	10/10
Resource Factor	0.9/1
Resource Constrainedness	0.5/0.75
Cumulative Resource produce interval	[8, 12]
Cumulative Resource consumption interval	[6, 12]

The objective function is to minimize total cost on remanufacturing project. Table 4.4 and 4.5 shows the result of possible mode selection in step one for the 18 instances used in this paper. Table 4.6 shows virtual precedence relation considering resource consumption.

The result for project bonus is shown in Fig 4.9. Since project duration and cumulative resource cost are the first to be minimized, heuristic method provides shorter project duration compared to embedded solver in normal case. For few instances, when cumulative resource inventory cost is high, a trade-off between cumulative resource inventory and project duration may be made to minimize object value in step 2. Averagely, heuristic method provides a schedule with 47% more bonus than normal method, which is efficient when dealing with time-sensitive projects.

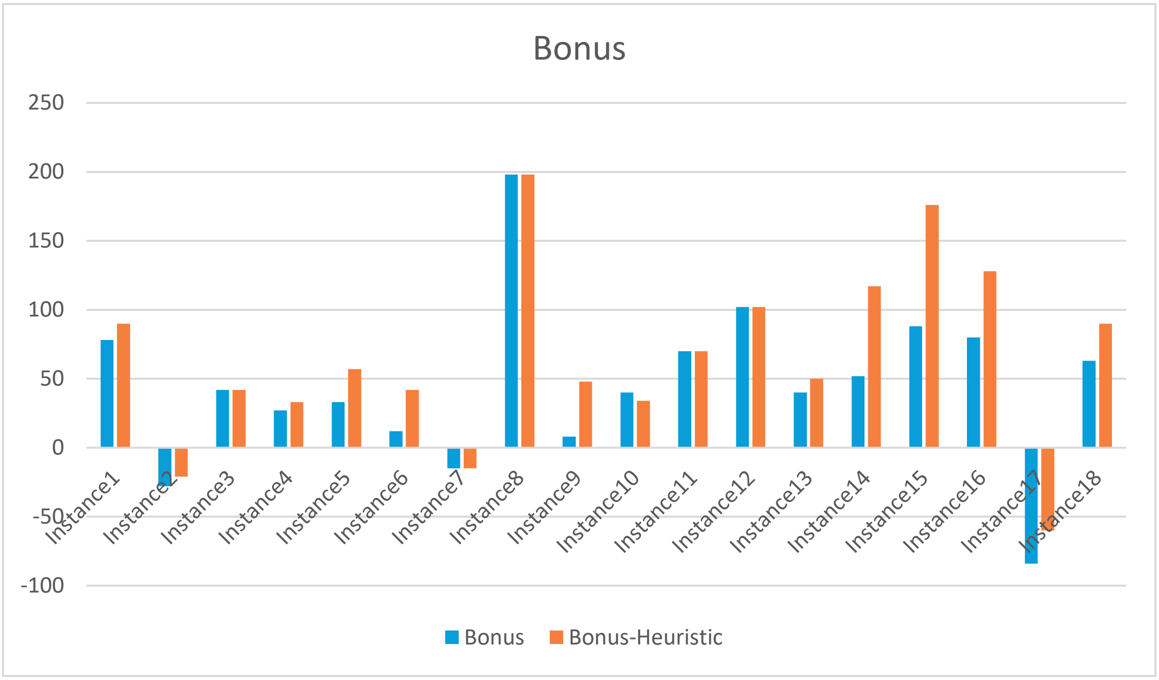


Fig 4.10 Bonus

Fig 4.10 show inventory cost for cumulative resource. For projects on a single high-value product, cumulative resource is usually recycled, inspected, and remanufactured at the very beginning of project when disassemble product external structure, then reused in later assembly. Since recycled components requires further inspection before reuse, it is not sure how much storage should be allocated for remanufactured resource in advance. Besides, some of the components may deteriorate after recycled. Therefore, it is efficient for remanufacturing project to minimize cumulative resource inventory level in step 2. On average, heuristic method shows 0.9% percent gap in cumulative resource cost. However, the average inventory time before reuse from heuristic method is slightly lower than normal method.

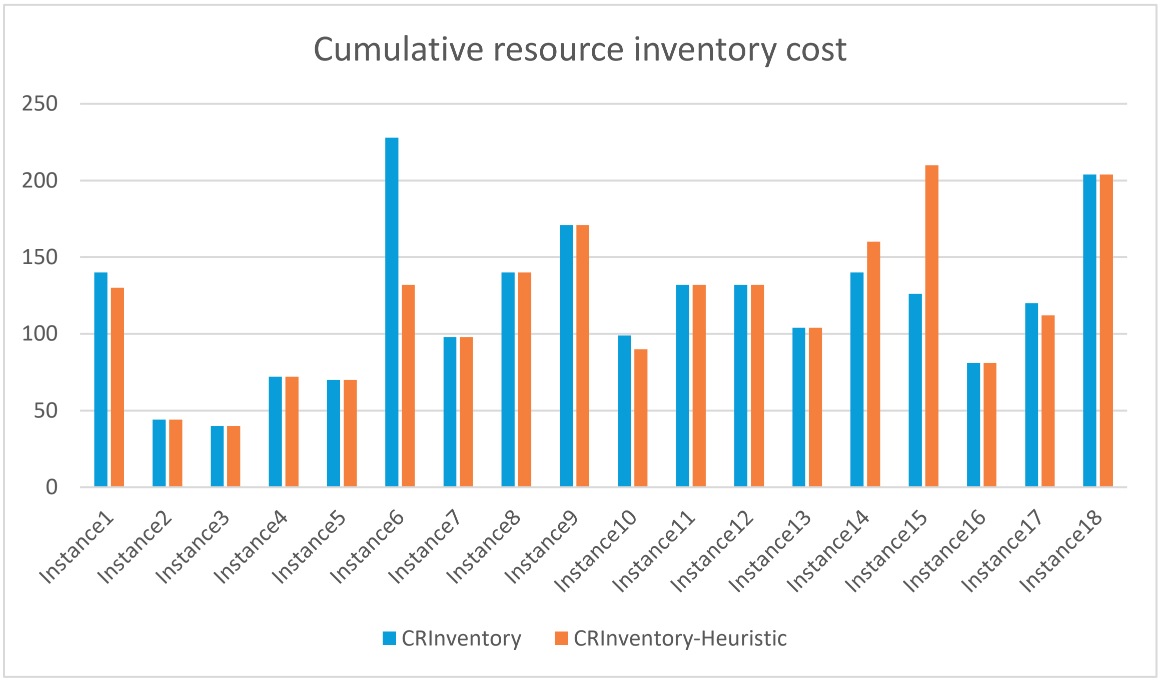


Fig 4.11 Cumulative resource inventory cost

Non-renewable resource cost consists of three parts: purchase cost, inventory cost, and order cost. Purchase cost is related to activity duration. Inventory denotes the how long materials will be stored before use. Order cost is a fixed number whenever an order for non-renewable resource is placed. In Fig 4.11 non-renewable resource purchase cost for each instance is shown. On average, heuristic method incurs 21% more purchase cost than optimal solution.

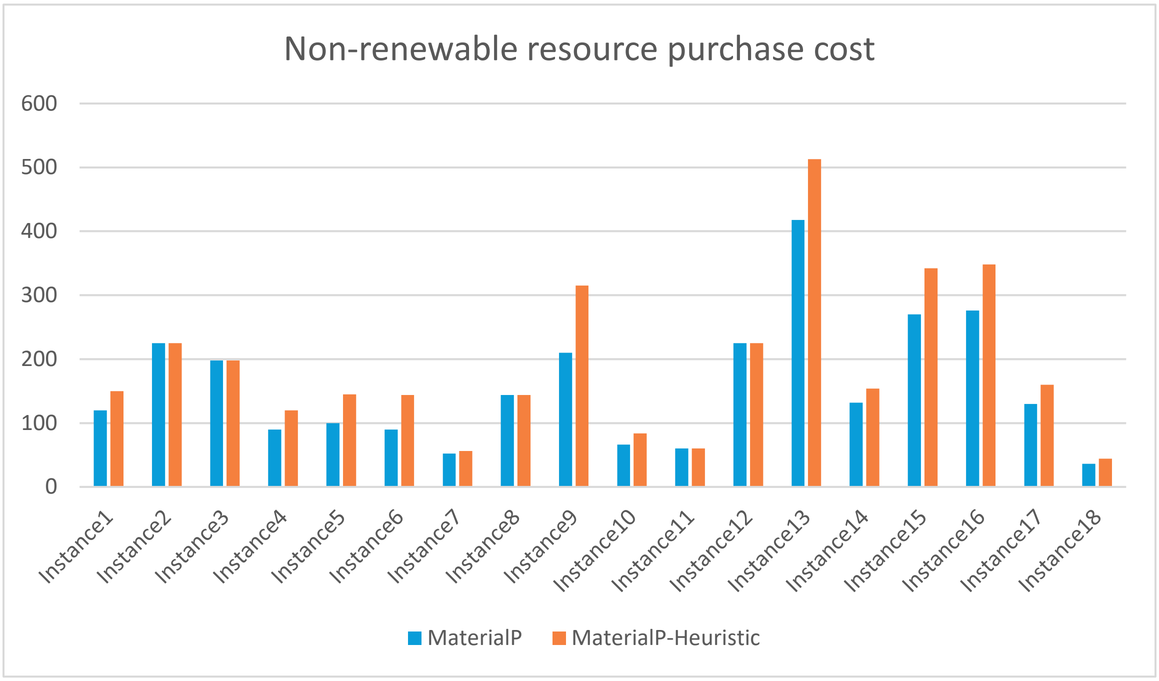


Fig 4.12 Non-renewable resource purchase cost

As for non-renewable inventory cost, heuristic method shows 21% percent gap compared to optimal solutions, which is similar to purchase cost. The inventory gap comes from higher volume of material purchase. The average storage time before process for each unit of non-renewable resource is similar to optimal method, which means heuristic method is capable of reducing redundant inventory.

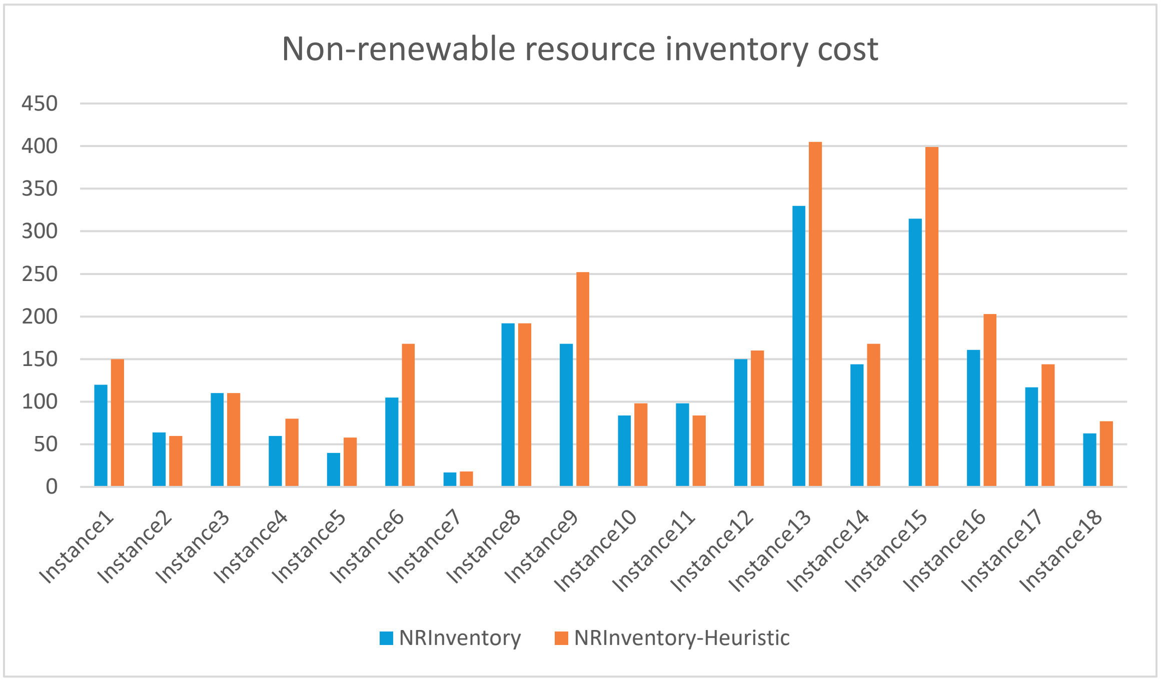


Fig 4.13 Non-renewable resource inventory cost

Order cost is independent from volume of purchased material. Purchase large quantity of non-renewable resource incurs low order cost while gives heavy pressure on inventory management, while frequent order generates additional process fee without reduce inventory cost. The result for order cost is shown is Fig 4.13. The result from heuristic method is 10% higher averagely than optimal method.

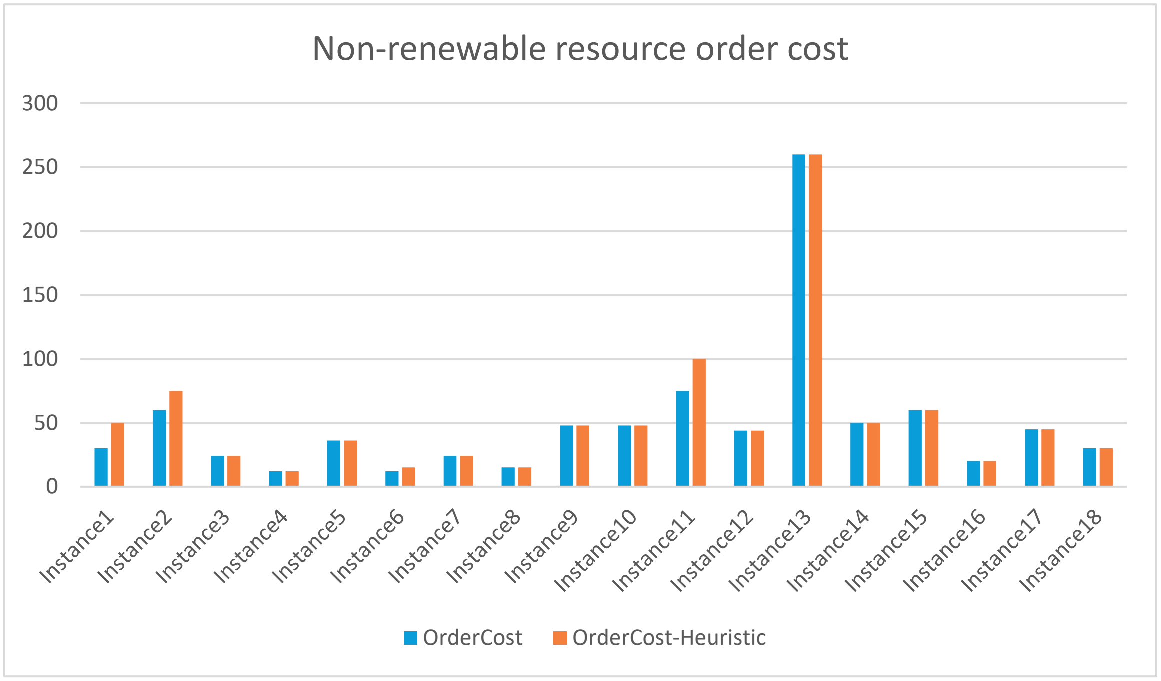


Fig 4.14 Non-renewable resource order cost

The results in Fig 4.14 show 10% gap between heuristic method and normal method. In practice situation, project schedule requires frequent adjustment with respect to actual progress, which requires a fast method to generate a schedule for reference. In Table 4.11 we can see heuristic method takes 20 seconds in average to reach a suboptimal solution with satisfied accuracy, consider that remanufacturing activities need to be scheduled.

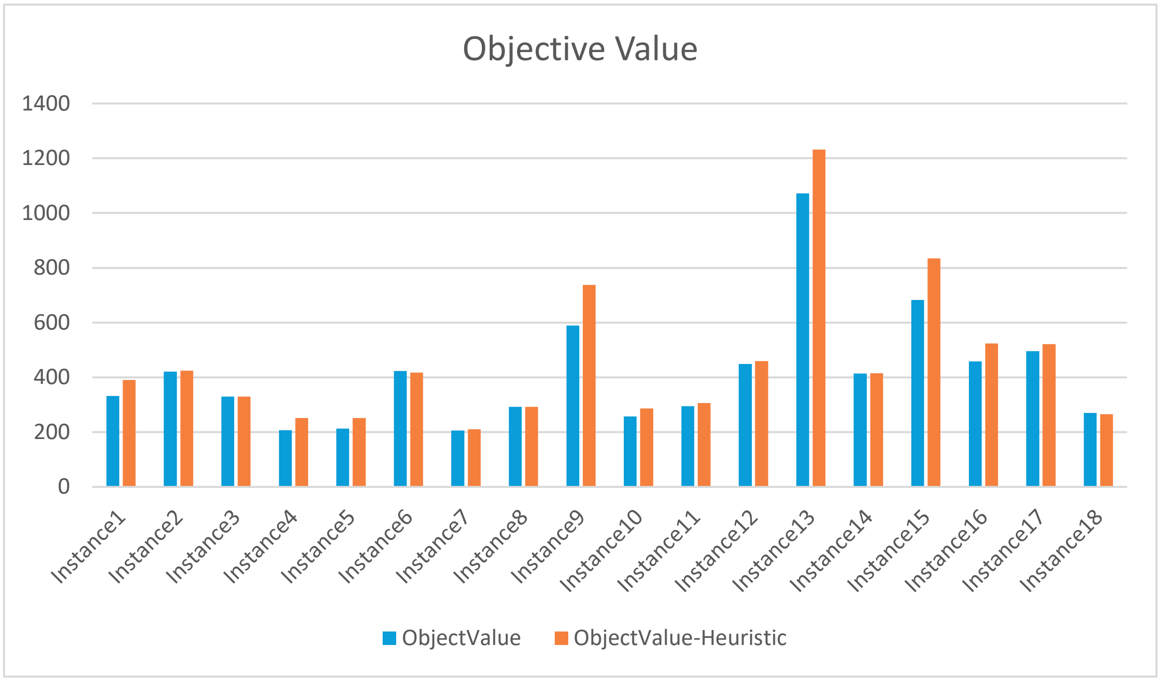


Fig 4.15 Total cost

Table 4.11 Cost and time consumption for each instance

	B	BH	CI	CH	MP	MH	NI	NIH	O	OH	Cost	CH	TimeH	Gap
I1	78	90	140	130	120	150	120	150	30	50	332	390	0:12	17%
I2	-28	-21	44	44	225	225	64	60	60	75	421	425	0:07	1%
I3	42	42	40	40	198	198	110	110	24	24	330	330	0:07	0%
I4	27	33	72	72	90	120	60	80	12	12	207	251	0:11	21%
I5	33	57	70	70	100	145	40	58	36	36	213	252	0:13	18%
I6	12	42	228	132	90	144	105	168	12	15	423	417	0:52	-1%
I7	-15	-15	98	98	52	56	17	18	24	24	206	211	0:05	2%
I8	198	198	140	140	144	144	192	192	15	15	293	293	0:07	0%
I9	8	48	171	171	210	315	168	252	48	48	589	738	0:25	25%
I10	40	34	99	90	66	84	84	98	48	48	257	286	0:07	11%
I11	70	70	132	132	60	60	98	84	75	100	295	306	0:06	4%
I12	102	102	132	132	225	225	150	160	44	44	449	459	0:19	2%
I13	40	50	104	104	418	513	330	405	260	260	1072	1232	0:17	15%
I14	52	117	140	160	132	154	144	168	50	50	414	415	0:53	0%
I15	88	176	126	210	270	342	315	399	60	60	683	835	0:37	22%
I16	80	128	81	81	276	348	161	203	20	20	458	524	0:45	14%
I17	-84	-60	120	112	130	160	117	144	45	45	496	521	0:44	5%
I18	63	90	204	204	36	44	63	77	30	30	270	265	0:10	-2%

The result for all 18 instances is shown in Table 4.11. For 18 tested instance, CPLEX solver takes 2 hours to obtain a known-optimal solution, while heuristic method takes 20 seconds averagely to get a suboptimal solution, which is acceptable in practical situation.

4.5 Summary

In this chapter, a three-phase heuristic method for solving RCPSP with remanufacturing activity is proposed. In first stage, considering actual cumulative resource consumption and production, a set of virtual precedence relations based on resource balance is generated. In step 2, a scheduling problem aims to minimize the sum of cumulative resource inventory cost and bonus is solved. Original precedence and resource-based precedence are used here. In step 3, a schedule for non-renewable resource purchase and inventory is generated to minimize non-renewable material related cost based on activity schedule from step 2.

The heuristic method takes acceptable computation time to obtain a suboptimal schedule. For 18 test instances, heuristic method obtains solution with 10% gap from CPLEX solver on average. The result also shows that proposed heuristic method provides better solution when time becomes important, with 47% better result on bonus than CPLEX solver. Since non-renewable resource related costs are highly relevant, proposed heuristic method is not efficient on products which cannot recover important core components.

Chapter 5 Summary and Future work

In this thesis, we consider remanufacturing activity in multi-mode resource constrained project scheduling problems. The disassembly structure is introduced into material ordering in the form of resource production and consumption. The integration of remanufacturing and RCPSP provides an effective provides an effective model for project scheduling. An integer programming model is developed to demonstrate this problem where remanufacturing activity can produce resource needed by normal activity. This thesis aims to provide a practical method for scheduling a project considering cumulative resource usage on an easy-accessible computing system such as personal computer while maintains relative accurate result to known optimal solution in large-scale project. The problem is a NP-hard problem which cannot be solved by exact methods. To do so, a three-phase heuristic method is developed in order to minimize total project cost under penalty and bonus policy.

Though commercial software provides accessible solution for models, it takes long time to solve a model, which is not acceptable in practical situation. Besides, off-shelf model requires excessive knowledge in order to adjust parameters for quick convergence to better solution, and usually cost much to obtain a commercial license. In large scale projects, off-shelf optimization methods do not guarantee a global optimal solution. The proposed three-phase heuristic method can quickly reach a suboptimal solution in practical case.

A set of 18 RanGen1 generated instances is used to validate the efficiency of proposed model.

As all instances use same cumulative resource information, remanufactured resource production and consumption combinations are enumerated in Excel to generate virtual precedence which reduced complexity in step 2. The second step use original precedence and virtual precedence to schedule the project in order to minimize sum of cumulative resource, bonus, and penalty. In last stage, a non-renewable resource order schedule is generated based on activity schedule from previous step to minimize non-renewable resource related cost. The second step and third step are implemented in ILOG CPLEX 12.8.

For future work, as the second step still use classic scheduling function, different meta-heuristic method can be used to accelerate the second phase. Besides, a solution pool can be generated to store all partial schedule in step 2 within acceptable gap range, which increase the chance to improve suboptimal schedule in step 3.

Reference

[1] Demeulemeester, E.L. and Herroelen, W.S., *Project scheduling: a research handbook* (Vol. 49). Springer Science & Business Media, 2006.

[2] Bounds, G., The last word on project management. *IIE solutions*, *30*(11), pp.41-44, 1998.

[3] Ilgin, M.A. and Gupta, S.M., *Remanufacturing modeling and analysis*. CRC Press, pp. 4, 2012.

[4] Clark, W., *The Gantt chart: A working tool of management*. Ronald Press Company, 1922.

[6] Brennan, M., PERT and CPM: a selected bibliography. *Council of Planning Librarians*, 1968.

[7] Kolisch, R., Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, *14*(3), pp.179-192, 1996.

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

[9] Hartmann, S. and Kolisch, R., Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *127*(2), pp.394-407, 2000.

[10] Alcaraz, J. and Maroto, C., A robust genetic algorithm for resource allocation in project scheduling. *Annals of operations Research*, *102*(1), pp.83-109, 2001.

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

[12] Hartmann, S., A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, *49*(5), pp.433-448, 2002.

[13] Bouleimen, K.L.E.I.N. and Lecocq, H.O.U.S.N.I., A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European journal of operational research*, *149*(2), pp.268-281, 2003.

[14] Möhring, R.H., Schulz, A.S., Stork, F. and Uetz, M., Solving project scheduling problems by minimum cut computations. *Management science*, *49*(3), pp.330-350, 2003.

[15] Guide Jr, V.D.R., Souza, G.C. and Van Der Laan, E., Performance of static priority rules for shared facilities in a remanufacturing shop with disassembly and reassembly. *European Journal of Operational Research*, *164*(2), pp.341-353, 2005.

[16] Debels, D., De Reyck, B., Leus, R. and Vanhoucke, M., A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, *169*(2), pp.638-653, 2006.

[17] Zhu, G., Bard, J.F. and Yu, G., A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, *18*(3), pp.377-390, 2006.

[18] Chtourou, H. and Haouari, M., A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & industrial engineering*, *55*(1), pp.183-194, 2008.

[19] Valls, V., Ballestin, F. and Quintanilla, S., A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *185*(2), pp.495-508.

[20] Debels, D., De Reyck, B., Leus, R. and Vanhoucke, M., 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, *169*(2), pp.638-653, 2008.

[21] Jarboui, B., Damak, N., Siarry, P. and Rebai, A., A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, *195*(1), pp.299-308, 2008.

[22] Mendes, J.J., Gonçalves, J.F. and Resende, M.G., A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & operations research*, *36*(1), pp.92-109, 2009.

[23] Hartmann, S. and Briskorn, D., A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, *207*(1), pp.1-14, 2010.

[24] Zheng, X.L. and Wang, L., A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Systems with Applications*, *42*(15-16), pp.6039-6049, 2015.

[25] Arkhipov, D., Battaïa, O. and Lazarev, A., An efficient pseudo-polynomial algorithm for finding a lower bound on the makespan for the Resource Constrained Project Scheduling Problem. *European Journal of Operational Research*, *275*(1), pp.35-44, 2019.

[26] Hartmann, S. and Briskorn, D., A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, *207*(1), pp.1-14, 2010.

[27] Mika, M., Waligora, G. and Węglarz, J., Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research*, *187*(3), pp.1238-1250, 2008.

[28] Drezet, L.E. and Billaut, J.C., A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics*, *112*(1), pp.217-225, 2008.

[29] Neumann, K. and Schwindt, C., Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, *56*(3), pp.513-533, 2003.

[30] Zoraghi, N., Shahsavar, A., Abbasi, B. and Van Peteghem, V., Multi-mode resource-constrained project scheduling problem with material ordering under bonus–penalty policies. *Top*, *25*(1), pp.49-79, 2017.

[31] Salewski, F., Schirmer, A. and Drexl, A., Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. *European Journal of Operational Research*, *102*(1), pp.88-110, 1997.

[32] Ahn, T. and Erenguc, S.S., The resource constrained project scheduling problem with multiple crashable modes: a heuristic procedure. *European Journal of Operational Research*, *107*(2), pp.250-259, 1998.

[33] Li, H. and Womer, K., Modeling the supply chain configuration problem with resource constraints. *International Journal of Project Management*, *26*(6), pp.646-654, 2008.

[34] Tareghian, H.R. and Taheri, S.H., A solution procedure for the discrete time, cost and quality tradeoff problem using electromagnetic scatter search. *Applied mathematics and computation*, *190*(2), pp.1136-1145, 2007.

[35] Tiwari, V., Patterson, J.H. and Mabert, V.A., Scheduling projects with heterogeneous resources to meet time and quality objectives. *European Journal of Operational Research*, *193*(3), pp.780-790, 2009.

[36] Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W. and Vanhoucke, A.M., New computational results on the discrete time/cost trade-off problem in project networks. *Journal of the operational research society*, *49*(11), pp.1153-1163, 1998.

[37] Drexl, A., Nissen, R., Patterson, J.H. and Salewski, F., Progen/πx–an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research*, *125*(1), pp.59-72, 2000.

[38] Vanhoucke, M., Work continuity constraints in project scheduling. *Journal of Construction Engineering and Management*, *132*(1), pp.14-25, 2006.

[39] Bartusch, M., Möhring, R.H. and Radermacher, F.J., Scheduling project networks with resource constraints and time windows. *Annals of operations Research*, *16*(1), pp.199-240, 1988.
[40] Özdamar, L., Ulusoy, G. and Bayyigit, M., A heuristic treatment of tardiness and net present value criteria in resource constrained project scheduling. *International Journal of Physical Distribution & Logistics Management*, 1998.

[41] Neumann, K., Schwindt, C. and Zimmermann, J., *Recent results on resource-constrained project scheduling with time windows: Models, solution methods, and applications*. Inst. für Wirtschaftstheorie und Operations-Research, 2002.

[42] Klein, R. and Scholl, A., PROGRESS: Optimally solving the generalized resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, *52*(3), pp.467-488, 2000.
[43] Gupta, S.M. and Taleb, K.N., Scheduling disassembly. *The International Journal of Production Research*, *32*(8), pp.1857-1866, 1994.

[44] Lee, D.H. and Xirouchakis, P., A two-stage heuristic for disassembly scheduling with assembly product structure. *Journal of the Operational Research Society*, *55*(3), pp.287-297, 2004.

[45] Taleb, K.N., Gupta, S.M. and Brennan, L., Disassembly of complex product structures with parts and materials commonality. *Production Planning & Control*, *8*(3), pp.255-269, 1997.

[46] Lee, D.H., Kim, H.J., Choi, G. and Xirouchakis, P., Disassembly scheduling: integer programming models. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *218*(10), pp.1357-1372, 2004.

[47] Kim, H.J., Lee, D.H., Xirouchakis, P. and Kwon, O.K., A branch and bound algorithm for disassembly scheduling with assembly product structure. *Journal of the Operational Research Society*, *60*(3), pp.419-430, 2009.

[48] Kim, H.J., Lee, D.H. and Xirouchakis, P., A Lagrangean heuristic algorithm for disassembly scheduling with capacity constraints. *Journal of the Operational Research Society*, *57*(10), pp.1231-1240, 2006.

[49] Kim, J.G., Jeon, H.B., Kim, H.J., Lee, D.H. and Xirouchakis, P., Disassembly scheduling with capacity constraints: minimizing the number of products disassembled. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *220*(9), pp.1473-1481, 2006.

[50] Jeon, H.B., Kim, J.G., Kim, H.J., and Lee, D.H., A two-stage heuristic for disassembly scheduling with capacity constraints. *Management Science and Financial Engineering*, *12*(1), pp.95-112, 2006.

[51] Kim, H.J., Lee, D.H. and Xirouchakis, P., Two-phase heuristic for disassembly scheduling with multiple product types and parts commonality. *International Journal of Production Research*, *44*(1), pp.195-212, 2006.

[52] Herroelen, W.S., Resource-constrained project scheduling—the state of the art. *Journal of the Operational Research Society*, *23*(3), pp.261-275, 1972.
[53] Demeulemeester, E., Vanhoucke, M. and Herroelen, W., RanGen: A random network generator for activity-on-the-node networks. *Journal of scheduling*, *6*(1), pp.17-38, 2003.

[54] Cooper, D.F., Heuristics for scheduling resource-constrained projects: An experimental investigation. *Management Science*, *22*(11), pp.1186-1194, 1976.

[55] Patterson, J.H., Project scheduling: The effects of problem structure on heuristic performance. *Naval Research Logistics Quarterly*, *23*(1), pp.95-123, 1976.

[56] Pritsker, A.A.B., Waiters, L.J. and Wolfe, P.M., Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, *16*(1), pp.93-108, 1969.

Appendix A

```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: tszha
 * Creation Date: Aug 14, 2020 at 5:48:23 PM
 *********************************************/

 using CP;

 int T = ... ; //Time period numbers
 int M = ... ; //Manufacturing activity number
 int R = ... ; //Remanufacturing activity number
 int S = ... ; //Renewable resource number
 int N = ... ; //Non-renewable resource number
 int C = ... ; //Cumulative resource number
 int D = ... ; //Activity mode number

 range TP = 0..T-1; //Time set
 range AM = 0..M-1; //Manufacturing activity set
 range AR = 0..R-1; //Remanufacturing activity set
 range RS = 0..S-1; //Renewable resource set
 range RN = 0..N-1; //Non-renewable resource set
 range RC = 0..C-1; //Cumulative resource set
 range MI = 0..D-1; //Mode set

 int PN = ...; // Penalty for late finish per day
 int BN = ...; // Bonus for early finish per day
 int DDL = ...; // Due date for project
 int MN = ...; // A large number

 int O[RN] = ...; // Order cost for non-renewable resource n
 int P[RN] = ...; // Purchase cost for each unit of NR resource n
 int H[RN] = ...; // Holding cost for NR n per unit period
 int L[RC] = ...; // Holding cost for CR c per unit period

 int RR[RS] = ...; // Max capacity of renewable resource s
 int DL[RN] = ...; // Delivery lead time of NR resource n
 int d[AM][MI] = ...; // Duration of normal activity i operate in mode m
 int r[AM][RS][MI] = ...; // Renewable resource s consumption when i in mode m
 int t[AM][RN][MI] = ...; // Non-renewable resource s consumption when i in mode m
 int u[AM][RC][MI] = ...; // Cumulative resource s consumption when i in mode m
 int dr[AR][MI] = ...; // Duration of Remanufacturing activity i operate in mode m
 int rr[AR][RS][MI] = ...; // Renewable resource s consumption when e in mode m
 int vr[AR][RC][MI] = ...; // Cumulative resource s production when e in mode m

 int esm[AM] = ...; // Earliest start time of normal activity i
 int lsm[AM] = ...; // Latest start time of normal activity i
 int esr[AR] = ...; // Earliest start time of remanufacturing activity i
 int lsr[AR] = ...; // Latest start time of remanufacturing activity i

 dvar boolean st[AM][MI][TP]; // 1 if normal activity i starts in mode m at time t
 dvar boolean rt[AR][MI][TP]; // 1 if remanufacturing activity e starts in mode m at
time t
 dvar boolean o[RN][TP]; // 1 if an order is placed for resource n at time t
 dvar int+ il[RN][TP]; // inventory level of non-renewable resource n at time t
 dvar int+ ic[RC][TP]; // invenroty level of cumulative resource c at time t
 dvar int+ no[RN][TP]; // Non-renewable resource n order number at time t

 tuple Precedence {int pred; int succ; }

 //{Precedence}

 {Precedence} PredN = ...; // Precedence set for normal activities
 {Precedence} PredR = ...; // Precedence set for remanufacturing activities



 dexpr int OrderCost = sum(rn in RN, tp in TP) O[rn]* o[rn][tp];
 dexpr int MaterialCost = sum(rn in RN, tp in TP) P[rn]* no[rn][tp];
 dexpr int NRInventoryCost = sum(rn in RN, tp in TP) H[rn]* il[rn][tp];
 dexpr int CRInventoryCost = sum(rc in RC, tp in TP) L[rc]* ic[rc][tp];
 dexpr int Penalty = sum(mi in MI, tp in TP: ((tp>=DDL+1) && (tp<=T))) PN *(tp-
DDL)*st[M-1][mi][tp];
 dexpr int Bonus = sum(mi in MI, tp in TP: ((tp>=esm[M-1]) && (tp<=DDL-1))) BN *(DDL-
tp)*st[M-1][mi][tp];

execute {
        cp.param.TimeLimit = 3600;
}

 minimize OrderCost+ MaterialCost+ NRInventoryCost+ CRInventoryCost+ Penalty- Bonus;

 subject to

 {
    Constraint1: forall (<i,j> in PredN) //Normal activity precedence constraint
        sum(mi in MI, tp in TP: (tp >=esm[i] && tp<=lsm[i])) st[i][mi][tp]*
(tp+d[i][mi])<= sum(mi in MI, tp in TP: (tp>= esm[j]&& tp<= lsm[j])) st[j][mi][tp]*
tp;

    Constraint2: forall (<i,j> in PredR) //Remanufacturing activity precedence
constraint
        sum(mi in MI, tp in TP: (tp >=esr[i] && tp<=lsr[i])) rt[i][mi][tp]*
(tp+dr[i][mi])<= sum(mi in MI, tp in TP: (tp>= esr[j]&& tp<= lsr[j])) rt[j][mi][tp]*
tp;

    Constraint3: forall (rs in RS, tp in TP) // Renewable resource capacity
constraint
        sum(i in AM, mi in MI, tm in TP: (tm <=tp && tm >=tp-d[i][mi]+1))
(st[i][mi][tm]* r[i][rs][mi])+ sum(j in AR, mi in MI, tr in TP: (tr >=tp-dr[j][mi]+1
&& tr<=tp)) (rt[j][mi][tr]* rr[j][rs][mi])<= RR[rs];

    Constraint4: forall (rn in RN, tp in TP: tp>=1) // Non-renewable resource
inventory level constraint(no initial)
        il[rn][tp] == il[rn][tp-1] + no[rn][tp-DL[rn]] - sum(mi in MI, i in AM)
t[i][rn][mi] * st[i][mi][tp];

    Constraint5: forall (rn in RN, tp in TP: tp>=1) // Non-renewable resource
consumption constraint(same day production cannot be used)
        sum(mi in MI, i in AM) t[i][rn][mi] * st[i][mi][tp] <= il[rn][tp-1];

    Constraint6: forall (rc in RC, tp in TP: tp>=1) // Cumulative resource
consumption constriant
        ic[rc][tp] == ic[rc][tp-1] + sum(mi in MI, e in AR, tr in TP: (tr == maxl(tp-
dr[e][mi], 0))) vr[e][rc][mi] * rt[e][mi][tr] - sum(mi in MI, j in AM) u[j][rc][mi] *
st[j][mi][tp];
```

```
    Constraint7: forall (rc in RC, tp in TP: tp>=1) // Cumulative resource
consumption constraint
        sum(mi in MI, i in AM) u[i][rc][mi] * st[i][mi][tp] <= ic[rc][tp-1];

    Constraint8: forall (i in AM) // Once and only once constraint for manufacturing
activity
        sum(mi in MI, tp in TP) st[i][mi][tp] == 1;

    Constraint9: forall (i in AR) // Once and only once constraint for
remanufacturing activity
        sum(mi in MI, tp in TP) rt[i][mi][tp] == 1;

    Constraint10: forall (rn in RN, tp in TP) // Large number Constriant
        no[rn][tp] <= o[rn][tp]*MN;

    Constraint11: forall (rn in RN) // Initial inventory level constriant
        il[rn][0] == 0;

    Constraint12: forall (rn in RN, tp in TP) // Positive integer inventory
constraint
        il[rn][tp] >= 0;

    Constraint13: forall (rc in RC) // Initial inventory level constriant
        ic[rc][0] == 0;

    Constraint14: forall (rc in RC, tp in TP) // Positive integer inventory
constraint
        ic[rc][tp] >= 0;

    Constraint15: forall (rc in RC, tp in TP) // Material order non-negative
constraint
        no[rc][tp] >= 0;

    Constraint16: sum (mi in MI, tp in TP) rt[0][mi][tp] == 1; //

    Constraint17: sum (mi in MI, tp in TP) rt[R-1][mi][tp]* tp <= sum (mi in MI, tp
in TP)st[M-1][mi][tp]* tp ; // R2 must finish before M6

    Constraint18: sum (mi in MI, tp in TP) rt[0][mi][tp]* tp >= 1; // R1 must start
after time 1
```

```
    Constraint19: st[0][0][1] == 1;

    Constraint20: sum (tp in TP) st[M-1][0][tp] == 1;

    Constraint21: forall(am in AM)
        sum(mi in MI, tp in TP: (tp >=esm[am] && tp<=lsm[am])) st[am][mi][tp] == 1;

    Constraint22:forall(ar in AR)
        sum(mi in MI, tp in TP: (tp >=esr[ar] && tp<=lsr[ar])) rt[ar][mi][tp] == 1;

}
```

Appendix B

```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: tszha
 * Creation Date: Dec 3, 2020 at 6:11:48 PM
 *********************************************/

using CP;

int T = ... ; //Time period numbers
int M = ... ; //Manufacturing activity number
int R = ... ; //Remanufacturing activity number
int S = ... ; //Renewable resource number
int N = ... ; //Non-renewable resource number
int C = ... ; //Cumulative resource number
int D = ... ; //Activity mode number

range TP = 0..T-1; //Time set
range AM = 0..M-1; //Manufacturing activity set
range AR = 0..R-1; //Remanufacturing activity set
range RS = 0..S-1; //Renewable resource set
range RN = 0..N-1; //Non-renewable resource set
range RC = 0..C-1; //Cumulative resource set
range MI = 0..D-1; //Mode set

int PN = ...; // Penalty for late finish per day
int BN = ...; // Bonus for early finish per day
int DDL = ...; // Due date for project
int MN = ...; // A large number

int O[RN] = ...; // Order cost for non-renewable resource n
int P[RN] = ...; // Purchase cost for each unit of NR resource n
int H[RN] = ...; // Holding cost for NR n per unit period
int L[RC] = ...; // Holding cost for CR c per unit period

int RR[RS] = ...; // Max capacity of renewable resource s
int DL[RN] = ...; // Delivery lead time of NR resource n
```

```
 int d[AM][MI] = ...; // Duration of normal activity i operate in mode m
 int r[AM][RS][MI] = ...; // Renewable resource s consumption when i in mode m
 int t[AM][RN][MI] = ...; // Non-renewable resource s consumption when i in mode m
 int u[AM][RC][MI] = ...; // Cumulative resource s consumption when i in mode m
 int dr[AR][MI] = ...; // Duration of Remanufacturing activity i operate in mode m
 int rr[AR][RS][MI] = ...; // Renewable resource s consumption when e in mode m
 int vr[AR][RC][MI] = ...; // Cumulative resource s production when e in mode m

 int esm[AM] = ...; // Earliest start time of normal activity i
 int lsm[AM] = ...; // Latest start time of normal activity i
 int esr[AR] = ...; // Earliest start time of remanufacturing activity i
 int lsr[AR] = ...; // Latest start time of remanufacturing activity i

 dvar boolean st[AM][MI][TP]; // 1 if normal activity i starts in mode m at time t
 dvar boolean rt[AR][MI][TP]; // 1 if remanufacturing activity e starts in mode m at
time t
 //dvar boolean o[RN][TP]; // 1 if an order is placed for resource n at time t
 //dvar int+ il[RN][TP]; // inventory level of non-renewable resource n at time t
 dvar int+ ic[RC][TP] ; // invenroty level of cumulative resource c at time t
 dvar int+ no[RN][TP]; // Non-renewable resource n order number at time t

 tuple Precedence {int pred; int succ; }

 //{Precedence}

 {Precedence} PredN = ...; // Precedence set for normal activities
 {Precedence} PredR = ...; // Precedence set for remanufacturing activities



 //dexpr int OrderCost = sum(rn in RN, tp in TP) O[rn]* o[rn][tp]; // Total order
cost
 dexpr int MaterialCost = sum(rn in RN, tp in TP) P[rn]* no[rn][tp]; // Total non-
renewable resource purchase cost
 //dexpr int NRInventoryCost = sum(rn in RN, tp in TP) H[rn]* il[rn][tp]; // total
non-renewable material inventory cost
 dexpr int CRInventoryCost = sum(rc in RC, tp in TP) L[rc]* ic[rc][tp]; // total
cumulative resource inventory cost
 dexpr int Penalty = sum(mi in MI, tp in TP: ((tp>=DDL+1) && (tp<=T))) PN *(tp-
DDL)*st[M-1][mi][tp]; // Penalty
```

```
 dexpr int Bonus = sum(mi in MI, tp in TP: ((tp>=esm[M-1]) && (tp<=DDL-1))) BN *(DDL-
tp)*st[M-1][mi][tp]; // Bonus
 dexpr int Production[rc in RC] = sum(ar in AR, mi in MI, tp in TP) rt[ar][mi][tp]*
vr[ar][rc][mi];
 dexpr int Consumption[rc in RC] = sum(am in AM, mi in MI, tp in TP) st[am][mi][tp]*
u[am][rc][mi];



execute {
        cp.param.TimeLimit = 1800;
}

 minimize CRInventoryCost+ Penalty- Bonus+ MaterialCost;

 subject to

 {
   Constraint1: forall (<i,j> in PredN) //Normal activity precedence constraint
        sum(mi in MI, tp in TP: (tp >=esm[i] && tp<=lsm[i])) st[i][mi][tp]*
(tp+d[i][mi])<= sum(mi in MI, tp in TP: (tp>= esm[j]&& tp<= lsm[j])) st[j][mi][tp]*
tp;

   Constraint2: forall (<i,j> in PredR) //Remanufacturing activity precedence
constraint
        sum(mi in MI, tp in TP: (tp >=esr[i] && tp<=lsr[i])) rt[i][mi][tp]*
(tp+dr[i][mi])<= sum(mi in MI, tp in TP: (tp>= esr[j]&& tp<= lsr[j])) rt[j][mi][tp]*
tp;

   Constraint3: forall (rs in RS, tp in TP) // Renewable resource capacity
constraint
        sum(i in AM, mi in MI, tm in TP: (tm <=tp && tm >=tp-d[i][mi]+1))
(st[i][mi][tm]* r[i][rs][mi])+ sum(j in AR, mi in MI, tr in TP: (tr >=tp-dr[j][mi]+1
&& tr<=tp)) (rt[j][mi][tr]* rr[j][rs][mi])<= RR[rs];

   /*Constraint4: forall (rn in RN, tp in TP: tp>=1) // Non-renewable resource
inventory level constraint(no initial)
        il[rn][tp] == il[rn][tp-1] + no[rn][tp-DL[rn]] - sum(mi in MI, i in AM)
t[i][rn][mi] * st[i][mi][tp];
```

```
    Constraint5: forall (rn in RN, tp in TP: tp>=1) // Non-renewable resource 
consumption constraint(same day production cannot be used)
        sum(mi in MI, i in AM) t[i][rn][mi] * st[i][mi][tp] <= il[rn][tp-1];
    */
    Constraint6: forall (rc in RC, tp in TP: tp>=1) // Cumulative resource 
consumption constraint
        ic[rc][tp] == ic[rc][tp-1] + sum(mi in MI, e in AR, tr in TP: (tr == maxl(tp-
dr[e][mi], 0))) vr[e][rc][mi] * rt[e][mi][tr] - sum(mi in MI, j in AM) u[j][rc][mi] * 
st[j][mi][tp];

    Constraint7: forall (rc in RC, tp in TP: tp>=1) // Cumulative resource 
consumption constraint
        sum(mi in MI, i in AM) u[i][rc][mi] * st[i][mi][tp] <= ic[rc][tp-1];

    Constraint8: forall (i in AM) // Once and only once constraint for manufacturing 
activity
        sum(mi in MI, tp in TP) st[i][mi][tp] == 1;

    Constraint9: forall (i in AR) // Once and only once constraint for 
remanufacturing activity
        sum(mi in MI, tp in TP) rt[i][mi][tp] == 1;

    /*Constraint10: forall (rn in RN, tp in TP) // Large number Constriant for NR 
order
        no[rn][tp] <= o[rn][tp]*MN;

    Constraint11: forall (rn in RN) // Initial inventory level constriant
        il[rn][0] == 0;

    Constraint12: forall (rn in RN, tp in TP) // Positive integer inventory 
constraint
        il[rn][tp] >= 0;
    */
    Constraint13: forall (rc in RC) // Initial inventory level constriant
        ic[rc][0] == 0;

    Constraint14: forall (rc in RC, tp in TP) // Positive integer inventory 
constraint
        ic[rc][tp] >= 0;
    /*Constraint15: forall (rc in RC, tp in TP) // Material order non-negative
constraint
        no[rc][tp] >= 0;
    */

    Constraint16: sum (mi in MI, tp in TP) rt[0][mi][tp] == 1; //

    Constraint17: sum (mi in MI, tp in TP) rt[R-1][mi][tp]* tp <= sum (tp in TP)
st[M-1][0][tp]* tp ; // R2 must finish before M6

    Constraint18: sum (mi in MI, tp in TP) rt[0][mi][tp]* tp >= 1; // R1 must start
after time 1

    Constraint19: st[0][0][1] == 1;

    Constraint20: sum (tp in TP) st[M-1][0][tp] == 1;

    Constraint21: sum(mi in MI) st[1][mi][1] == 0;

    Constraint25: sum(mi in MI) st[2][mi][1] == 0;

    // Concept constraint

    Constarint22: forall (rc in RC) // Total production and consumption must be equal
        Production[rc] == Consumption[rc];

    //Constarint22: forall(rc in RC) // Total production and consumption must be
equal
    //  sum(mi in MI, e in AR, tp in TP) vr[e][rc][mi] * rt[e][mi][tp] == 12;

    Constraint23: forall(am in AM)
        sum(mi in MI, tp in TP: (tp >=esm[am] && tp<=lsm[am])) st[am][mi][tp] == 1;

    Constraint24: forall(ar in AR)
        sum(mi in MI, tp in TP: (tp >=esr[ar] && tp<=lsr[ar])) rt[ar][mi][tp] == 1;

 }
```

Appendix C

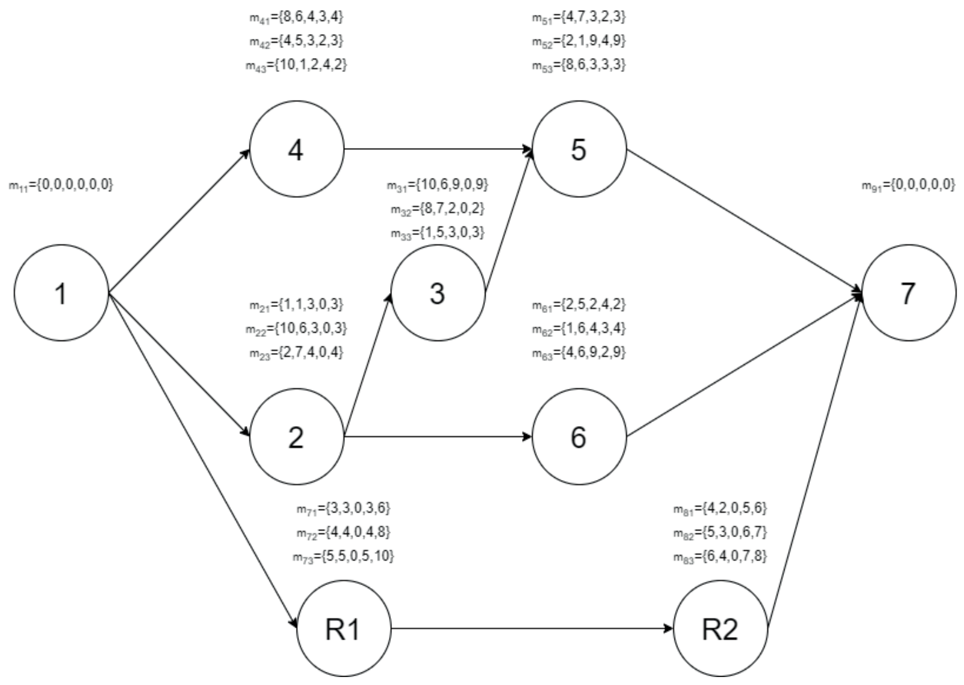
```
/*********************************************
 * OPL 12.8.0.0 Model
 * Author: tszha
 * Creation Date: Dec 13, 2020 at 12:06:44 PM
 *********************************************/

 using CP;

 int T = ... ; //Time period numbers
 int M = ... ; //Manufacturing activity number
 int R = ... ; //Remanufacturing activity number
 int S = ... ; //Renewable resource number
 int N = ... ; //Non-renewable resource number
 int C = ... ; //Cumulative resource number
 int D = ... ; //Activity mode number

 range TP = 0..T-1; //Time set
 range AM = 0..M-1; //Manufacturing activity set
 range AR = 0..R-1; //Remanufacturing activity set
 range RS = 0..S-1; //Renewable resource set
 range RN = 0..N-1; //Non-renewable resource set
 range RC = 0..C-1; //Cumulative resource set
 range MI = 0..D-1; //Mode set

 int PN = ...; // Penalty for late finish per day
 int BN = ...; // Bonus for early finish per day
 int DDL = ...; // Due date for project
 int MN = ...; // A large number

 int O[RN] = ...; // Order cost for non-renewable resource n
 int P[RN] = ...; // Purchase cost for each unit of NR resource n
 int H[RN] = ...; // Holding cost for NR n per unit period
 int L[RC] = ...; // Holding cost for CR c per unit period

 int RR[RS] = ...; // Max capacity of renewable resource s
 int DL[RN] = ...; // Delivery lead time of NR resource n
 int d[AM][MI] = ...; // Duration of normal activity i operate in mode m
 int r[AM][RS][MI] = ...; // Renewable resource s consumption when i in mode m
 int t[AM][RN][MI] = ...; // Non-renewable resource s consumption when i in mode m
 int u[AM][RC][MI] = ...; // Cumulative resource s consumption when i in mode m
 int dr[AR][MI] = ...; // Duration of Remanufacturing activity i operate in mode m
 int rr[AR][RS][MI] = ...; // Renewable resource s consumption when e in mode m
 int vr[AR][RC][MI] = ...; // Cumulative resource s production when e in mode m
 int AMD[AM][0..2] = ...;
 int ARD[AR][0..2] = ...;

 int esm[AM] = ...; // Earliest start time of normal activity i
 int lsm[AM] = ...; // Latest start time of normal activity i
 int esr[AR] = ...; // Earliest start time of remanufacturing activity i
 int lsr[AR] = ...; // Latest start time of remanufacturing activity i

 int st[AM][MI][TP] ; //1 if normal activity i starts in mode m at time t
 int rt[AR][MI][TP] ; //1 if remanufacturing activity e starts in mode m at time t

 dvar boolean o[RN][TP]; // 1 if an order is placed for resource n at time t
 //dvar int+ ic[RC][TP]; //invenroty level of cumulative resource c at time t
 dvar int+ il[RN][TP]; // inventory level of non-renewable resource n at time t
 dvar int+ no[RN][TP]; // Non-renewable resource n order number at time t

 tuple Precedence {int pred; int succ; }

 //{Precedence}

 {Precedence} PredN = ...; // Precedence set for normal activities
 {Precedence} PredR = ...; // Precedence set for remanufacturing activities




 dexpr int OrderCost = sum(rn in RN, tp in TP) O[rn]* o[rn][tp]; // Total order cost
 dexpr int MaterialCost = sum(rn in RN, tp in TP) P[rn]* no[rn][tp]; // Total non-
renewable resource purchase cost
 dexpr int NRInventoryCost = sum(rn in RN, tp in TP) H[rn]* il[rn][tp]; // total non-
renewable material inventory cost
 //dexpr int CRInventoryCost = sum(rc in RC, tp in TP) L[rc]* ic[rc][tp]; // total
cumulative resource inventory cost
```

```
 //dexpr int Penalty = sum(mi in MI, tp in TP: ((tp>=DDL+1) && (tp<=T))) PN *(tp-
DDL)*st[M-1][mi][tp]; // Penalty
 //dexpr int Bonus = sum(mi in MI, tp in TP: ((tp>=esm[M-1]) && (tp<=DDL-1))) BN
*(DDL- tp)*st[M-1][mi][tp]; // Bonus

execute {
        cp.param.TimeLimit = 1800;
}

execute{
    for (var am in AM){
        for (var mi in MI){
            for (var tp in TP){
                st[am][mi][tp] =0;
            }
        }
        st[AMD[am][0]][AMD[am][1]][AMD[am][2]] =1;
    };

    for (var ar in AR){
        for (var mi in MI){
            for (var tp in TP){
                rt[ar][mi][tp] =0;
            }
        }
        rt[ARD[ar][0]][ARD[ar][1]][ARD[ar][2]] =1;
    };
}

 minimize OrderCost+ MaterialCost+ NRInventoryCost;

 subject to

 {
    /*Constraint1: forall (<i,j> in PredN) //Normal activity precedence constraint
        sum(mi in MI, tp in TP: (tp >=esm[i] && tp<=lsm[i])) st[i][mi][tp]*
(tp+d[i][mi])<= sum(mi in MI, tp in TP: (tp>= esm[j]&& tp<= lsm[j])) st[j][mi][tp]*
tp;

    Constraint2: forall (<i,j> in PredR) //Remanufacturing activity precedence
constraint
```

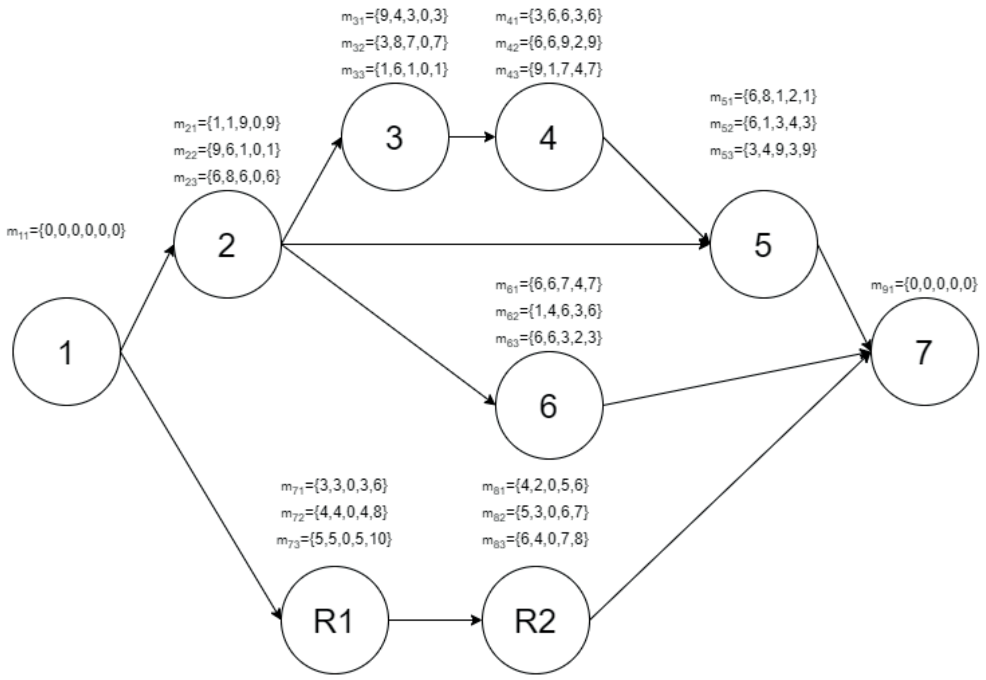
```
        sum(mi in MI, tp in TP: (tp >=esr[i] && tp<=lsr[i])) rt[i][mi][tp]*
(tp+dr[i][mi])<= sum(mi in MI, tp in TP: (tp>= esr[j]&& tp<= lsr[j])) rt[j][mi][tp]*
tp;

    Constraint3: forall (rs in RS, tp in TP) // Renewable resource capacity
constraint
        sum(i in AM, mi in MI, tm in TP: (tm <=tp && tm >=tp-d[i][mi]+1))
(st[i][mi][tm]* r[i][rs][mi])+ sum(j in AR, mi in MI, tr in TP: (tr >=tp-dr[j][mi]+1
&& tr<=tp)) (rt[j][mi][tr]* rr[j][rs][mi])<= RR[rs];
    */
    Constraint4: forall (rn in RN, tp in TP: tp>=1) // Non-renewable resource
inventory level constraint(no initial)
        il[rn][tp] == il[rn][tp-1] + no[rn][tp-DL[rn]] - sum(mi in MI, i in AM)
t[i][rn][mi] * st[i][mi][tp];

    Constraint5: forall (rn in RN, tp in TP: tp>=1) // Non-renewable resource
consumption constraint(same day production cannot be used)
        sum(mi in MI, i in AM) t[i][rn][mi] * st[i][mi][tp] <= il[rn][tp-1];

    /*Constraint6: forall (rc in RC, tp in TP: tp>=1) // Cumulative resource
consumption constriant
        ic[rc][tp] == ic[rc][tp-1] + sum(mi in MI, e in AR, tr in TP: (tr == maxl(tp-
dr[e][mi], 0))) vr[e][rc][mi] * rt[e][mi][tr] - sum(mi in MI, j in AM) u[j][rc][mi] *
st[j][mi][tp];

    Constraint7: forall (rc in RC, tp in TP: tp>=1) // Cumulative resource
consumption constraint
        sum(mi in MI, i in AM) u[i][rc][mi] * st[i][mi][tp] <= ic[rc][tp-1];

    Constraint8: forall (i in AM) // Once and only once constraint for manufacturing
activity
        sum(mi in MI, tp in TP) st[i][mi][tp] == 1;

    Constraint9: forall (i in AR) // Once and only once constraint for
remanufacturing activity
        sum(mi in MI, tp in TP) rt[i][mi][tp] == 1;
    */
    Constraint10: forall (rn in RN, tp in TP) // Large number Constriant
        no[rn][tp] <= o[rn][tp]*MN;

    Constraint11: forall (rn in RN) // Initial inventory level constriant
```

```
        il[rn][0] == 0;

    Constraint12: forall (rn in RN, tp in TP) // Positive integer inventory
constraint
        il[rn][tp] >= 0;

    /*Constraint13: forall (rc in RC) // Initial inventory level constriant
        ic[rc][0] == 0;

    Constraint14: forall (rc in RC, tp in TP) // Positive integer inventory
constraint
        ic[rc][tp] >= 0;
    */
    Constraint15: forall (rc in RC, tp in TP) // Material order non-negative
constraint
        no[rc][tp] >= 0;

    /*Constraint16: sum (mi in MI, tp in TP) rt[0][mi][tp] == 1; //

    Constraint17: sum (mi in MI, tp in TP) rt[R-1][mi][tp]* tp <= sum (mi in MI, tp
in TP)st[M-1][mi][tp]* tp ; // R2 must finish before M6

    Constraint18: sum (mi in MI, tp in TP) rt[0][mi][tp]* tp >= 1; // R1 must start
after time 1

    Constraint19: st[0][0][1] == 1;

    Constraint20: sum (tp in TP) st[M-1][0][tp] == 1;

    Constraint21: forall (am in AM)
        st[AMD[am][0]][AMD[am][1]][AMD[am][2]] == 1;

    Constraint22: forall (ar in AR)
        rt[ARD[ar][0]][ARD[ar][1]][ARD[ar][2]] == 1;
    */
```

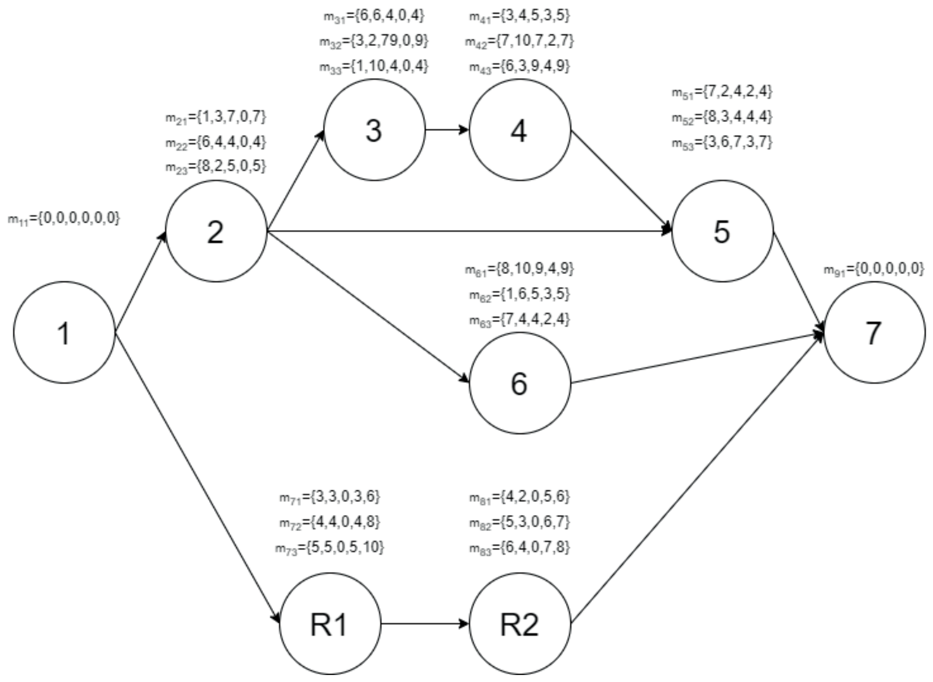
Appendix D



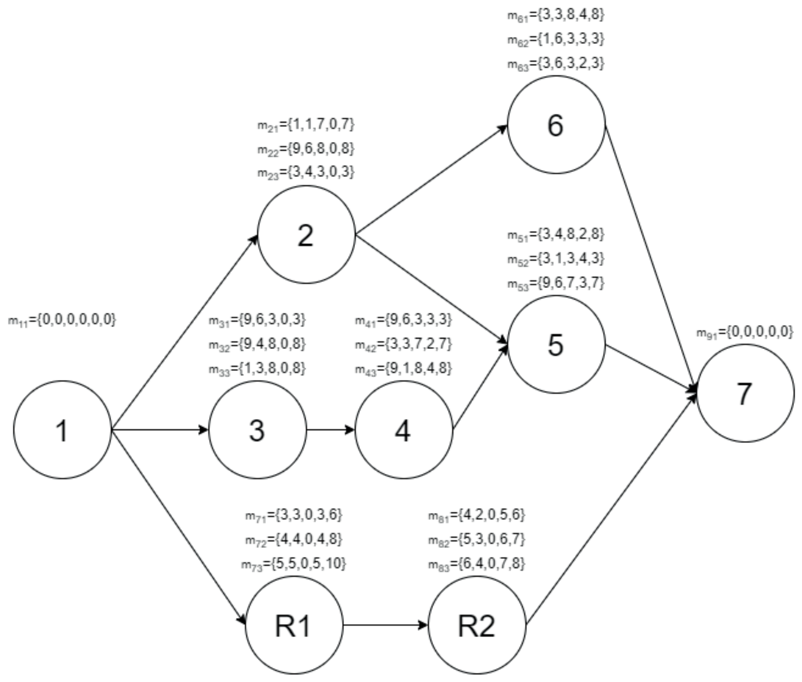
Instance 1



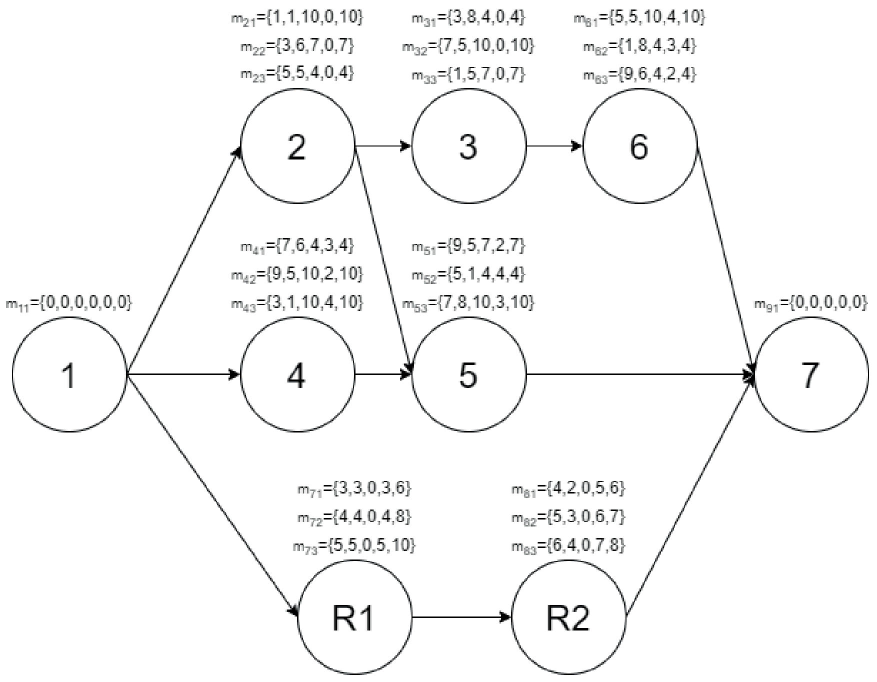
Instance 2



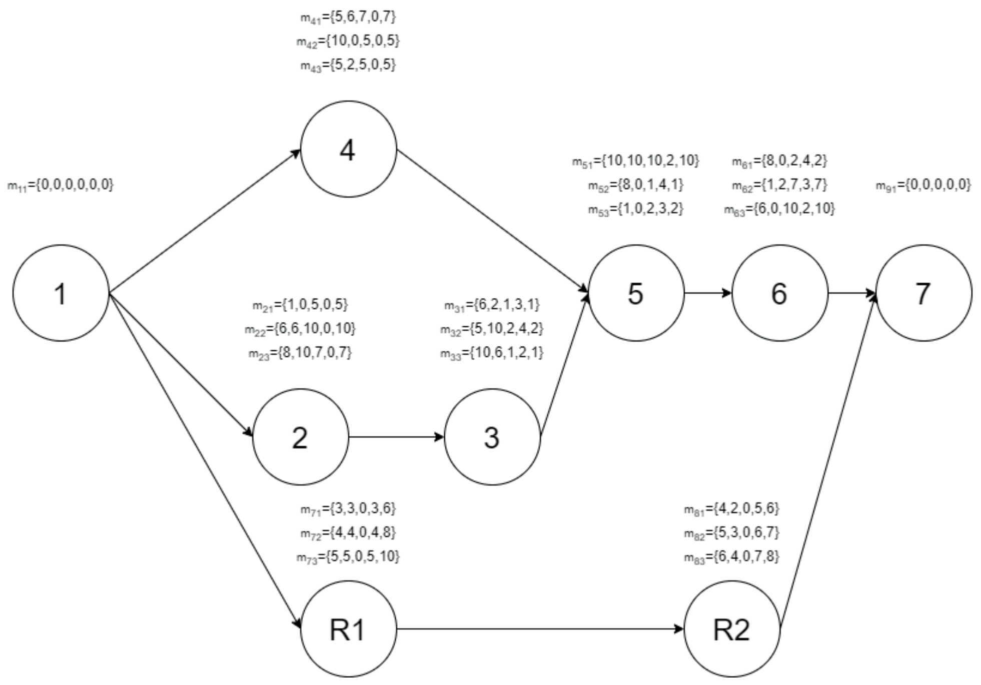
Instance 3



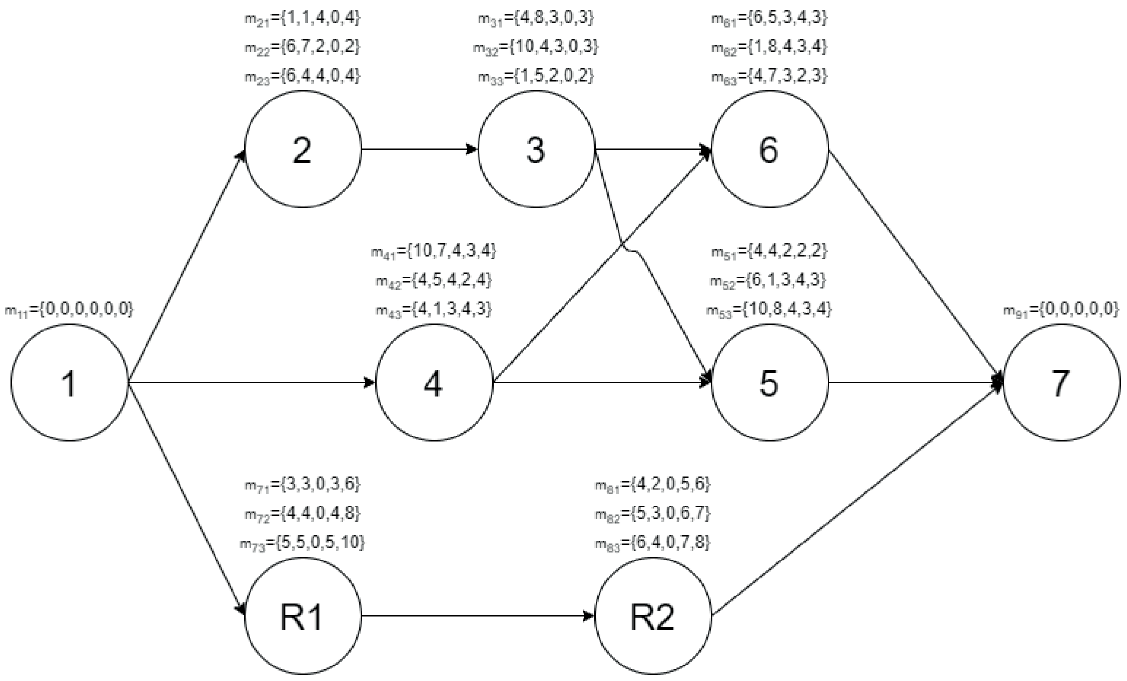
Instance 4



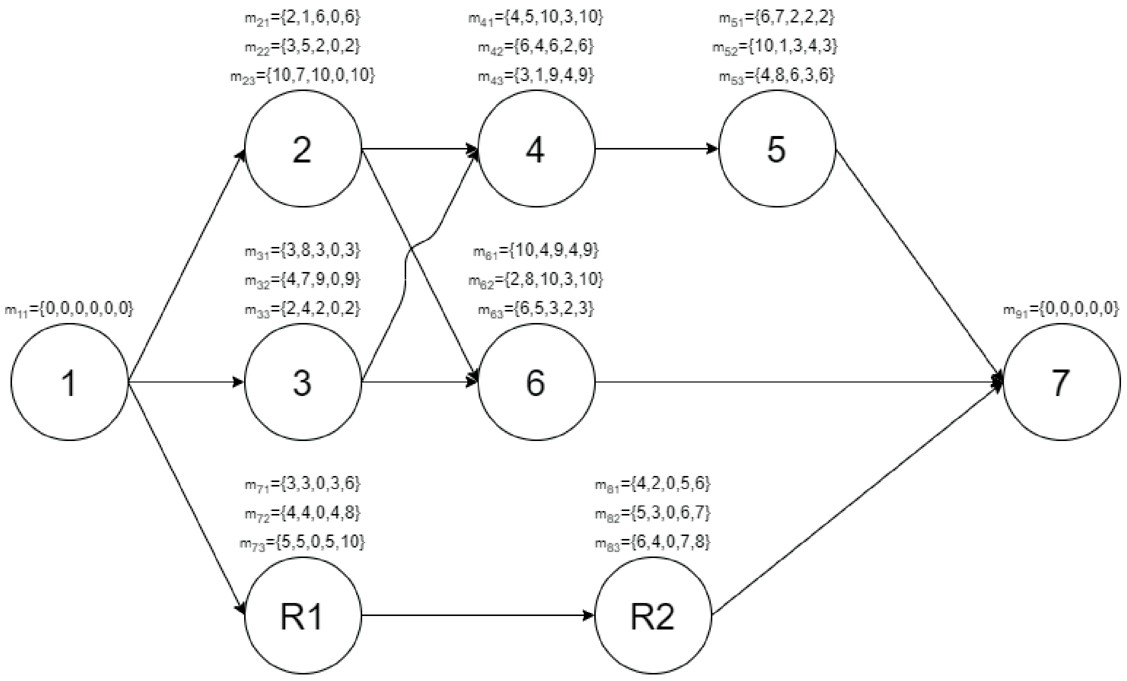
Instance 5



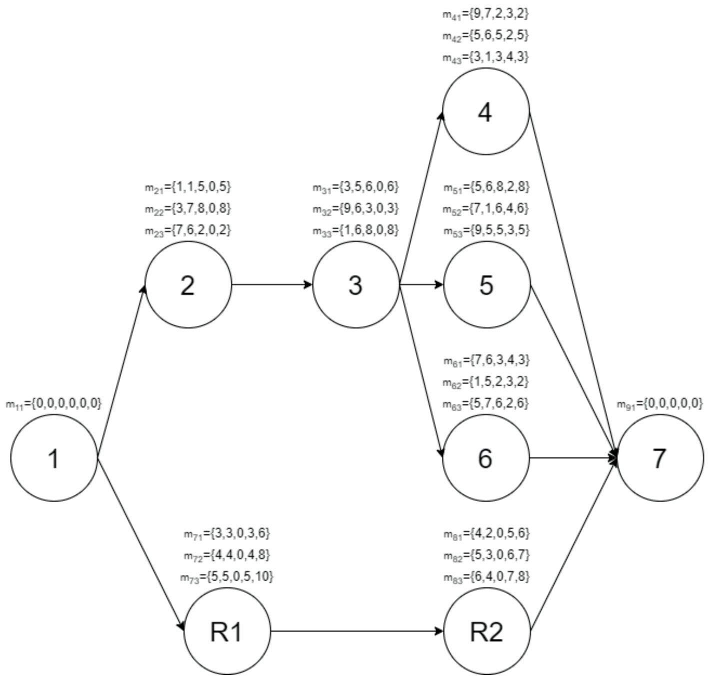
Instance 6



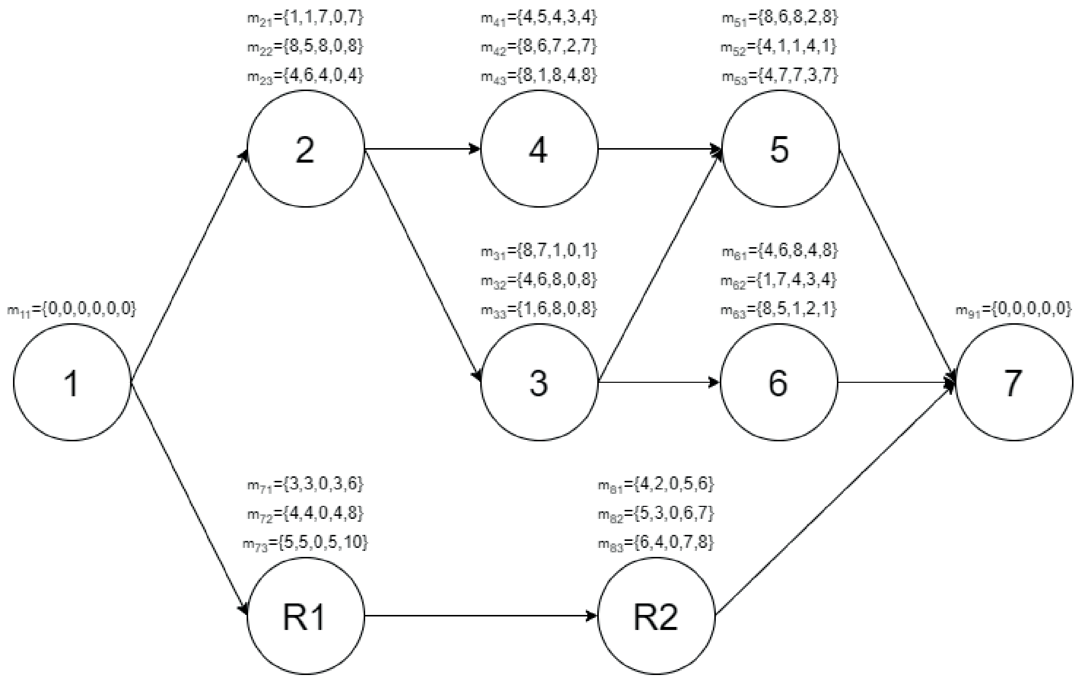
Instance 7



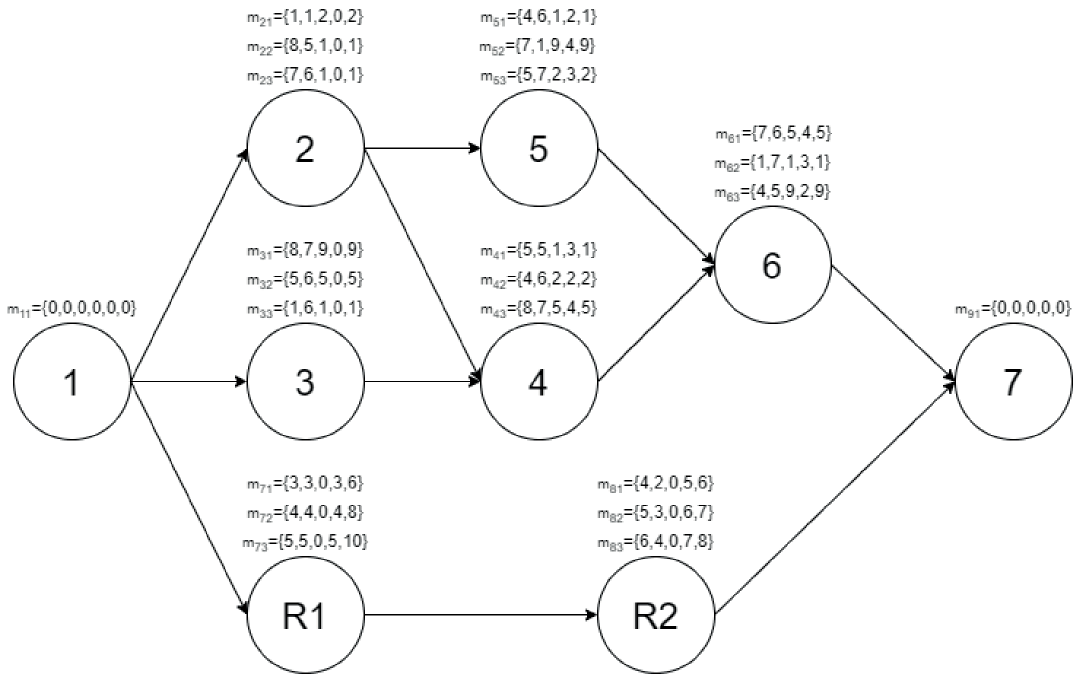
Instance 8



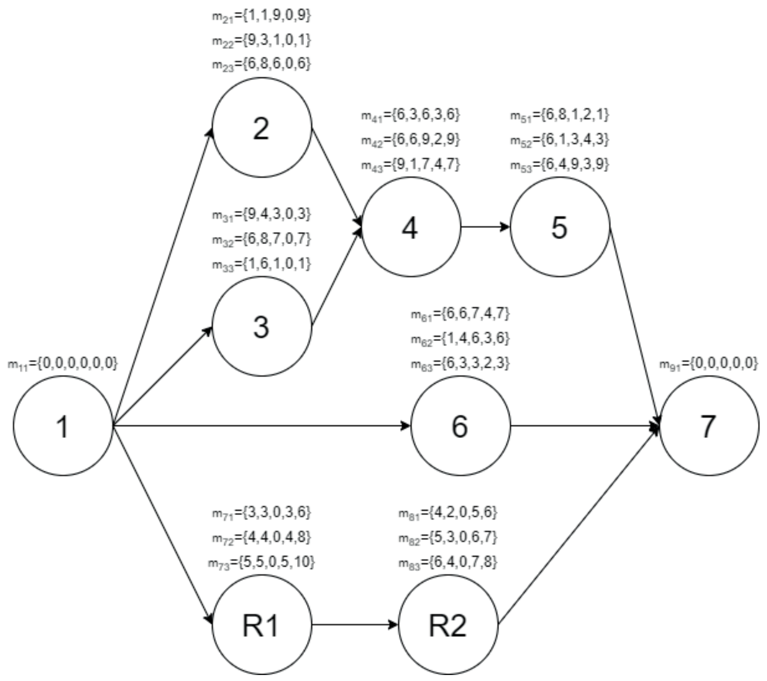
Instance 9



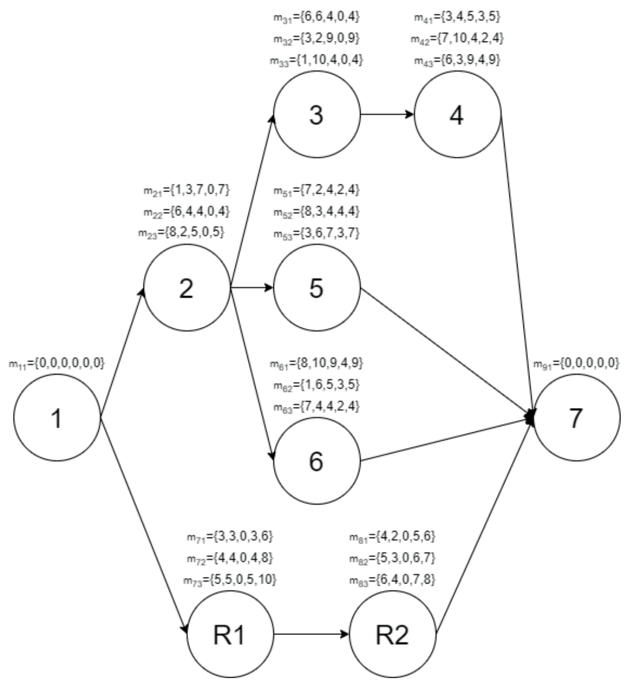
Instance 10


m₂₁={1,1,2,0,2}
m₂₂={8,5,1,0,1}
m₂₃={7,6,1,0,1}
m₅₁={4,6,1,2,1}
m₅₂={7,1,9,4,9}
m₅₃={5,7,2,3,2}
m₆₁={7,6,5,4,5}
m₆₂={1,7,1,3,1}
m₆₃={4,5,9,2,9}
m₃₁={8,7,9,0,9}
m₃₂={5,6,5,0,5}
m₃₃={1,6,1,0,1}
m₄₁={5,5,1,3,1}
m₄₂={4,6,2,2,2}
m₄₃={8,7,5,4,5}
m₁₁={0,0,0,0,0,0}
m₉₁={0,0,0,0,0}
1
2
3
4
5
6
7
m₇₁={3,3,0,3,6}
m₇₂={4,4,0,4,8}
m₇₃={5,5,0,5,10}
m₈₁={4,2,0,5,6}
m₈₂={5,3,0,6,7}
m₈₃={6,4,0,7,8}
R1
R2

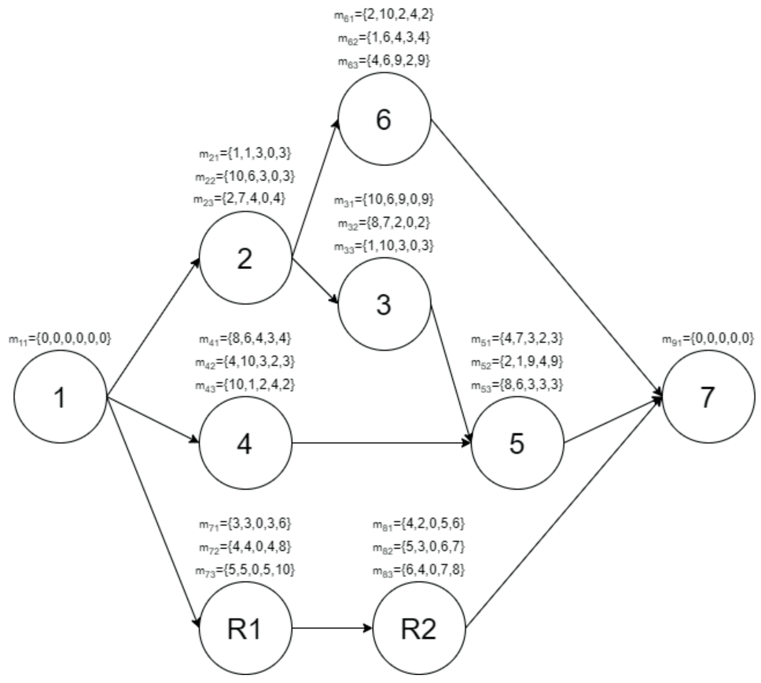
Instance 11


m₂₁={1,1,9,0,9}
m₂₂={9,3,1,0,1}
m₂₃={6,8,6,0,6}
m₄₁={6,3,6,3,6}
m₄₂={6,6,9,2,9}
m₄₃={9,1,7,4,7}
m₅₁={6,8,1,2,1}
m₅₂={6,1,3,4,3}
m₅₃={6,4,9,3,9}
m₃₁={9,4,3,0,3}
m₃₂={6,8,7,0,7}
m₃₃={1,6,1,0,1}
m₆₁={6,6,7,4,7}
m₆₂={1,4,6,3,6}
m₆₃={6,3,3,2,3}
m₁₁={0,0,0,0,0,0}
m₉₁={0,0,0,0,0}
1
2
3
4
5
6
7
m₇₁={3,3,0,3,6}
m₇₂={4,4,0,4,8}
m₇₃={5,5,0,5,10}
m₈₁={4,2,0,5,6}
m₈₂={5,3,0,6,7}
m₈₃={6,4,0,7,8}
R1
R2

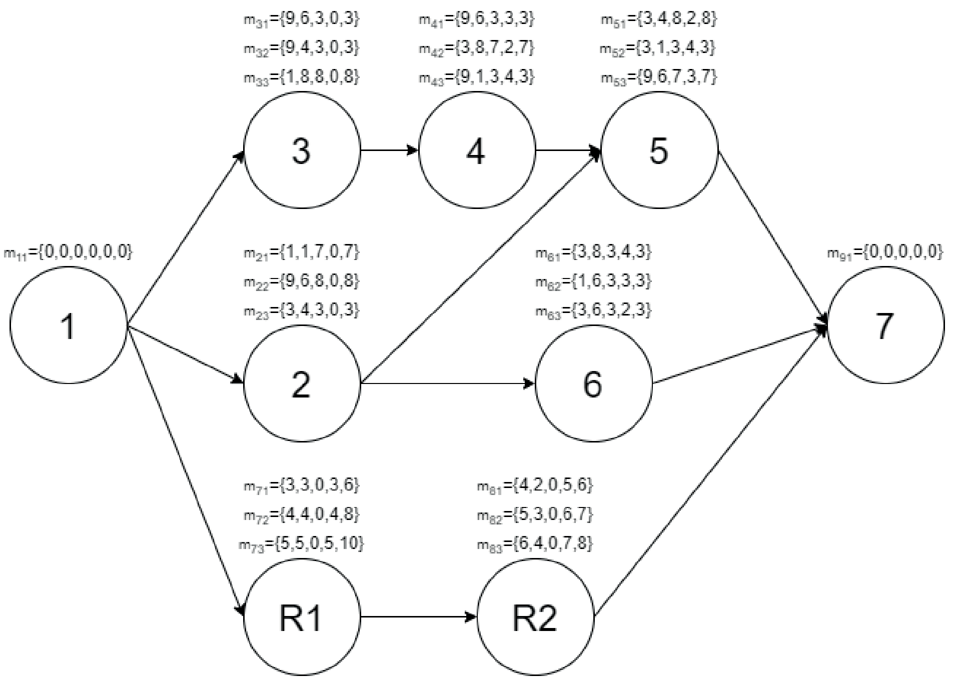
Instance 12



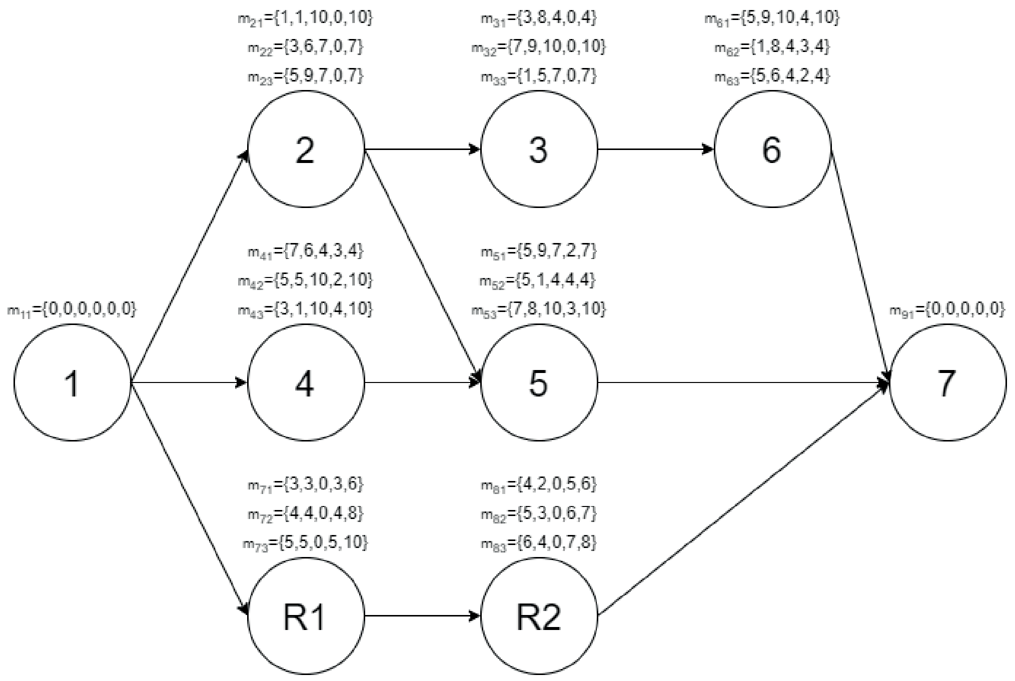
Instance 13



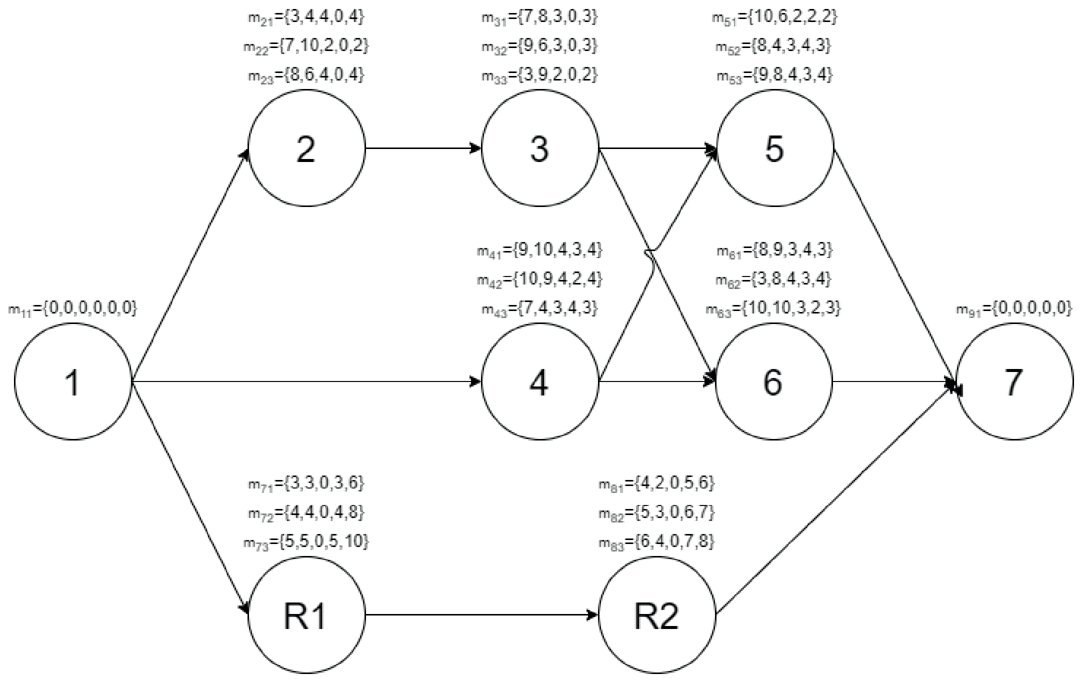
Instance 14



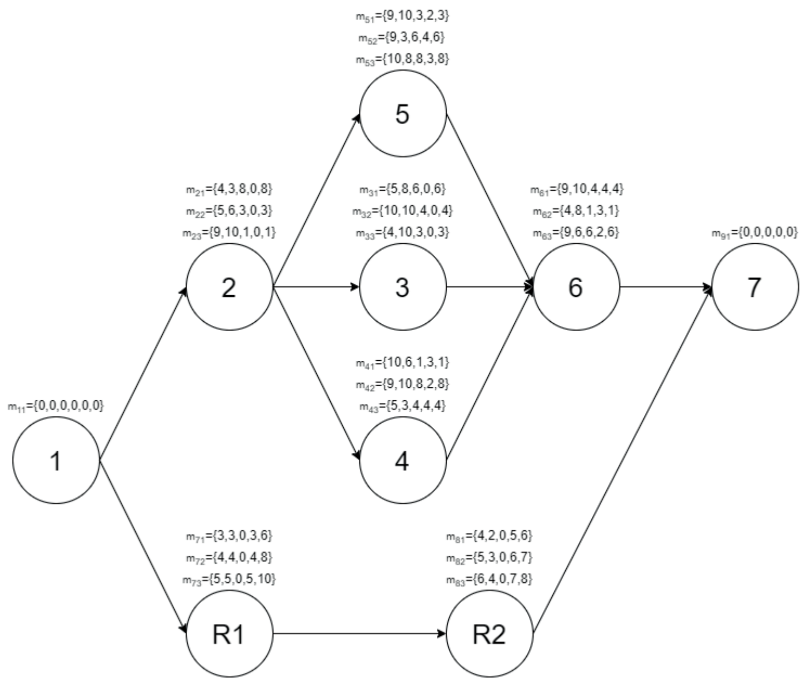
Instance 15



Instance 16



Instance 17



Instance 18

Appendix E

Instance #	1
Deadline	35
Penalty	5
Bonus	6
Order cost	10
NR resource price	10
NR resource inventory cost	10
CR resource inventory cost	10
Renewable resource caps	10, 10

Instance #	2
Deadline	20
Penalty	7
Bonus	3
Order cost	15
NR resource price	15
NR resource inventory cost	4
CR resource inventory cost	4
Renewable resource caps	10, 10

Instance #	3
Deadline	35
Penalty	4
Bonus	3
Order cost	6
NR resource price	9
NR resource inventory cost	5
CR resource inventory cost	5
Renewable resource caps	10, 10

Instance #	4
Deadline	35
Penalty	4
Bonus	3
Order cost	3
NR resource price	6
NR resource inventory cost	4
CR resource inventory cost	8
Renewable resource caps	10, 10

Instance #	5
Deadline	44
Penalty	4
Bonus	3
Order cost	9
NR resource price	5
NR resource inventory cost	2
CR resource inventory cost	7
Renewable resource caps	10, 10

Instance #	6
Deadline	41
Penalty	4
Bonus	3
Order cost	3
NR resource price	6
NR resource inventory cost	7
CR resource inventory cost	12
Renewable resource caps	10, 10

Instance #	7
Deadline	16
Penalty	15
Bonus	15
Order cost	8
NR resource price	4
NR resource inventory cost	1
CR resource inventory cost	7
Renewable resource caps	10, 10

Instance #	8
Deadline	42
Penalty	12
Bonus	9
Order cost	3
NR resource price	9
NR resource inventory cost	12
CR resource inventory cost	7
Renewable resource caps	10, 10

Instance #	9
Deadline	31
Penalty	7
Bonus	8
Order cost	12
NR resource price	15
NR resource inventory cost	12
CR resource inventory cost	19
Renewable resource caps	10, 10

Instance #	10
Deadline	37
Penalty	4
Bonus	2
Order cost	12
NR resource price	6
NR resource inventory cost	7
CR resource inventory cost	9
Renewable resource caps	10, 10

Instance #	11
Deadline	26
Penalty	11
Bonus	7
Order cost	25
NR resource price	12
NR resource inventory cost	14
CR resource inventory cost	12
Renewable resource caps	10, 10

Instance #	12
Deadline	39
Penalty	7
Bonus	6
Order cost	11
NR resource price	15
NR resource inventory cost	10
CR resource inventory cost	12
Renewable resource caps	10, 10

Instance #	13
Deadline	25
Penalty	12
Bonus	10
Order cost	20
NR resource price	19
NR resource inventory cost	15
CR resource inventory cost	13
Renewable resource caps	10, 10

Instance #	14
Deadline	31
Penalty	11
Bonus	13
Order cost	10
NR resource price	11
NR resource inventory cost	12
CR resource inventory cost	10
Renewable resource caps	10, 10

Instance #	15
Deadline	30
Penalty	19
Bonus	22
Order cost	15
NR resource price	18
NR resource inventory cost	21
CR resource inventory cost	14
Renewable resource caps	10, 10

Instance #	16
Deadline	40
Penalty	9
Bonus	8
Order cost	5
NR resource price	12
NR resource inventory cost	7
CR resource inventory cost	9
Renewable resource caps	10, 10

Instance #	17
Deadline	22
Penalty	12
Bonus	12
Order cost	9
NR resource price	10
NR resource inventory cost	9
CR resource inventory cost	8
Renewable resource caps	10, 10

Instance #	18
Deadline	30
Penalty	8
Bonus	9
Order cost	6
NR resource price	4
NR resource inventory cost	7
CR resource inventory cost	12
Renewable resource caps	10, 10