INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD • INDIA

Research and Publications

Exact Procedures for Non-Regular Measures of the Multi-Mode RCPSP

Madhukar Dayal
Sanjay Verma

W.P. No.2015-03-06
March 2015

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

Exact Procedures For Non-Regular Measures of the Multi-Mode RCPSP

Prof. Madhukar Dayal, Indian Institute of Management Indore[1]
Prof. Sanjay Verma, Indian Institute of Management Ahmedabad

Abstract

The multi-mode resource constrained project scheduling problem (MM RCPSP) is a generalization of the well-studied RCPSP. A literature review reveals applications of inexact heuristics or metaheuristics approaches for solving these problems, however, exact approaches are few and do not consider non-renewable resources, as well as, generalized cash inflows and outflows at every time period of an activity, as is the case in real World problem instances. We present two exact solution single-processor approaches: a breadth-first tree search procedure and a best-first monotone heuristic for solving these problem instances. The algorithms are thoroughly tested on problem instances using payment schedules generated for standard PSPLIB problem sets and results presented.

Keywords: project scheduling, net present value, non-regular measures, exact solutions, breadth-first tree search, best-first.

[1] This work was done as part of Doctoral thesis by Prof. Dayal at IIM Ahmedabad.

1. Introduction

The project scheduling problem (PSP) continues to be studied extensively for regular, as well as, non-regular measures of performance. A large number of researchers have developed and presented inexact solution approaches, primarily deploying metaheuristics procedures, for both the above types of objectives. We have presented breadth-first and best-first exact solution approaches for the regular measures of performance which study the objective of minimizing makespan. Both of these approaches can be readily altered to pursue other non-regular measures as well, such as, minimizing completion time or minimizing tardiness. In this paper we present new algorithms for the multi-mode multiple resource constrained project scheduling problem for maximization of the net present value (NPV). Other interesting non-regular measures can also be pursued with minimal alteration to the algorithms presented, for example, lateness, earliness, tardiness.

The maximization of NPV in scheduling projects is an attractive objective frequently faced by managers in real life situations. However, the evaluation of the alternatives is extremely complex due to its tedious calculations, even with the help of modern computing support. Hence, sub-optimal approaches, for example thumb rules, are typically used by managers. However, these rules are not guaranteed to yield an optimal solution.

The pursuit of maximization of net present value (NPV) as an objective has gained more momentum in the last two decades of project scheduling research, first as a single objective, and more recently as one of the objectives in bi-criteria or multi criteria optimization. We modify our regular measure algorithms to suite NPV as the non-regular measure of performance. The algorithm is generalizable to other non-regular measures with minor changes. NPV is the present value of all future cash inflows minus outflows, discounted at the given discount rate (β), implying that the time value of money is taken into consideration. We consider the problem in which positive or negative cash flows are associated with each activity's start, end, and all time periods of its duration in the mode that it is being performed. Additionally, we consider the payment of a bonus for early completion of the whole project for up to four unit time periods before the given due date. The involvement of a due date in a project prevents activities with negative net cash flows from being indefinitely delayed. In the model studied, we do not consider the imposition of a penalty for late completion of the project, however it can be easily included, for example,

by considering negative bonus for the last one or more time periods after the due date as a penalty.

2. Literature review

Exact solution approaches for the NPV objective have been studied by very few researchers. Though the single mode case for maximization of npv has been considered for exact solution, such as Icmeli and Erenguc (1996) and Dayanand and Padman (1997), and the multi-mode case with only renewable resources has been considered by Vanhoucke, Demeulemeester, and Herroelen (2001); no approach has considered problems with non-renewable resources for exact solutions. The scheduling of partially ordered activities under renewable resource constraints exactly to optimize non-regular performance measures has been studied by Dhavale, Verma, and Bagchi (2003) applying a best-first tree search to minimize the total weighted earliness-tardiness and NPV. The primary reason for lack of research in solving this problem appears to be the substantial rise in computational overhead, even with only renewable resources, restricting feasibility to problem instances of a small size using exact approaches. The interest in the problem is clearly visible considering the large number of metaheuristic approaches which have been presented for solving this problem especially since 2000 AD.

Among metaheuristic approaches, the multi-mode case with renewable and non-renewable resources and payments associated with progress of activities have been studied by Ulusoy, Funda and Sahin (2001) deploying a Genetic Algorithm (GA) approach and Waligora (2008) using Tabu search. Sabzehparvar and Seyed-Hosseini (2008) have used an MILP formulation to solve small instances of the multi-mode case exactly and considered an abridged version for approximate solutions to larger instances. Metaheuristic approaches dominate the research in this particular problem model.

In this paper, we first describe the problem studied including the problem's mathematical formulation. Next we discuss the pre-processing rules applied before solving a problem instance for the NPV objective. Thereafter, the Breadth-first NPV and Best-first NPV algorithms for net present value maximization are presented. Subsequently, the proof of optimality of these algorithms is provided and experimental observations are discussed. The feasibility and potential of multi objective optimization using the Breadth-first NPV

algorithm is discussed briefly before the concluding remarks.

3. Problem statement

The non-regular measure NPV problem can be stated as follows. A project consists of N activities, N > 2. An activity i in its mode j, a_{ij}, has an integer duration $p_{ij} \geq 0$, and a set P_i of predecessor activities. M distinct types of renewable resources are available, $M \geq 1$, and the requirements for each of the M renewable resources for a_{ij} are given by r_{ijm}. The total availability of each of the M types of renewable resources is constant and a unit of renewable resource cannot be allocated to more than one activity at a given time. Once allocated, a renewable resource is free for allocation to another activity only when the activity to which it is allocated has finished.

K distinct types of non-renewable resources are available, $K \geq 1$, and the requirements for each of the K non-renewable resources for all activities in each of its modes (a_{ij}) are given by l_{ijk}. The total availability of each of the K types of non-renewable resources, L_k, is given at the start of the project and the total consumption of each non-renewable resource by all activities cannot exceed its given availability. Further, once scheduled an activity cannot be pre-empted. The given due date of the project is DD.

The cash flow at the start of an activity a_i in its mode j is CF_{ij0}, and at its end is CF_{ijt+1}, where t is its duration in mode j. The cash flows associated with each time period of its duration in mode j are CF_{ijt}, $0 < t \leq d_{ij}$, d_{ij} being the duration of a_{ij} (activity i in its mode j). Without loss of generality, it is assumed that a cash flow associated with a unit time period of an activity's mode occurs at the end of that unit time period. At the end of any activity, thus, the cash flow occurs on two accounts, viz. (a) one that is associated with the last unit time period of the activity's duration, and (b) the stipulated cash flow at the completion of that activity. The starting, ending, and all interim cash flows for an activity's mode are known prior to the start of the project, and are independent of each other, and also of its cash flows in any other mode. Any cash flow may be positive or negative, and the considered rate of discount per time period is β. For brevity, we consider a single applicable rate of interest per unit time period, for both, the short and long time periods. The incorporation of different interest rates applicable for given time slabs can also be easily modeled in our algorithm.

Payment of bonus is modeled as an activity consuming no time and no resources and occurring before the dummy end activity of the project. The bonus for the last four time periods up to the due date are B_0, B_1, B_2, and B_3 ($B_0 > B_1 > B_2 > B_3$). Note that the bonus values may be negative, indicating penalty for late completion. Typically, the largest bonus is associated with the earliest conceived finish time of the project (such as the feasible optimal makespan), and consecutively reducing amounts with each next unit time period up to the due date. In the completion horizon of four time periods, as soon as a negative bonus is encountered (representing a penalty for late completion), all subsequent 'bonuses' (i.e. penalties) are expected to be negative, too, and sequentially larger negative values.

Thus, for each activity (a_i) in all of its modes, given: (a) its predecessor set of activities, P_i; (b) processing times of the activity in each of its modes j, p_{ij}; (c) requirements of the renewable resources in each mode, r_{ijm}, for all renewable resources (for $1 \leq m \leq M$); (d) requirements of the non-renewable resources in each mode, l_{ijk}, for all non-renewable resources (for $1 \leq k \leq K$); (e) the associated cash flows with each mode (CF_{ij0}, CF_{ijt} ($0 < t \leq d_{ij}$), and CF_{ijt+1}); (f) the due date, DD ; (g) the discounting rate per time period, β ; and (h) the applicable bonus for the project (B_0, B_1, B_2, and B_3); the *scheduling* problem is to determine the resource feasible modes and start times for each activity such that the NPV of the project is maximized within the due date.

3.1. Mathematical formulation

The MM-RCPSP for the NPV objective can be mathematically stated as follows:

$$\text{Max } G_i = \sum_{i=1}^{N} E_i \quad (1)$$

such that

$f_i - f_j \geq p_{im}$, $1 \leq j < i \leq N$, $j \in P_i$ (processing time constraints). (2)

$\sum_{j=1}^{M} r_{imj} \leq R_j$, $1 \leq j \leq M$, $0 \leq t \leq f_N$ (renewable resource usage constraints). (3)

$\sum_{k=1}^{L} l_{imk} \leq L_k$, $1 \leq k \leq L$, $0 \leq t \leq f_N$ (non-renewable resource usage constraints). (4)

$f_N \leq DD$ (due date constraint) (5)

i, j, k, m, t, L, M, R > 0 (non-negativity constraints). (6)

i, j, k, m, t, L, M, R $\in$ I (Integrality constraints). (7)

The objective is to maximize the net present value of total cash flows from all activities, such that:

(i) The finishing time of an activity i processed in its mode m should be more than or equal to the finishing time of any of its predecessors plus the processing time of activity i in mode m.

(ii) At every integer time instant t, the total amount of any renewable resource deployed in all activities in progress at that time instant, in their respectively assigned modes, should not exceed the availability of that renewable resource.

(iii) At any time instant t, the total amount of any non-renewable resource consumed by activities completed and activities in progress, in their respectively assigned modes, should not exceed the total availability of that non-renewable resource.

(iv) $f_N \leq DD$, i.e. the finish time of last activity should be within the given due date (Note that due dates are not assigned in problems studied by us for makespan objective in Dayal and Verma (2014)).

4. Single-processor breadth-first NPV algorithm

We use a tree-traversal mechanism substantially similar to the one given in Dayal and Verma (2014) with alterations for non-regular measures. The project bonus (or penalty) at the completion of the project is modeled as a cash flow associated with an end activity consuming no resources and no time. Thus, the dummy end activity, representing the actual completion of the project, now has a single predecessor activity consuming no resources and time, and involving no other cash flows except the bonus cash flow, which is determined by the completion time of the project.

4.1. Pre-processing rules

Similar to the pre-processing rules for regular measures, the NPV problem instances are also pre-processed through these rules to filter resource infeasible modes of an activity and remove redundant non-renewable resource(s) which are available in abundance. However, identification and removal of inferior or identical modes requires additional consideration with respect to the associated cash flows (see Dayal and Verma (2014) for complete description of pre-processing).

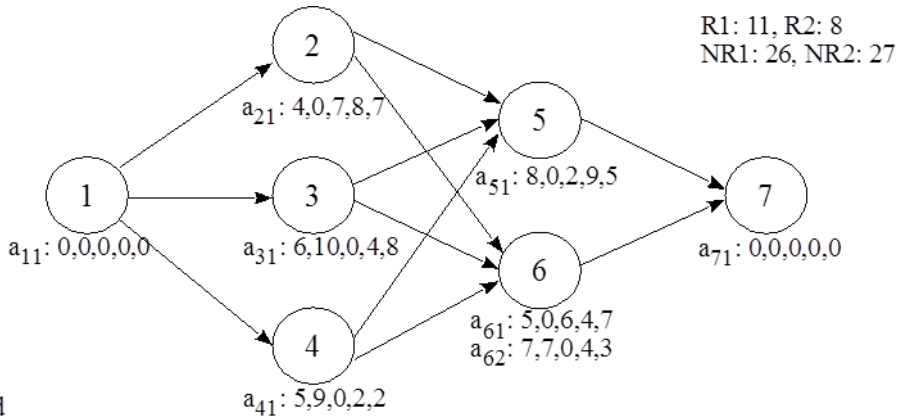
The consideration of an inferior mode now involves the cash flows too, as even if a mode

yields an inferior net cash flow with respect to another mode when both are considered starting at t = 0, the situation may not be same for these modes for some start time other than zero (i.e., t > 0) with values of β strictly greater than zero. The pre-processing rule can also not be applied based only on discounted terminal cash flows (DCF_{ij}) of the two modes. Two modes of an activity with different durations are incomparable, as the longer mode would have associated cash flows for time periods which do not exist in the shorter mode. A change in the value of applicable interest rate β, too, may alter the superiority of two modes of an activity mutually. We consider a fixed value of β given at the start and applicable throughout the project's duration. Thus, a mode of an activity (mode j_1) is superior to another mode of the activity (mode j_2), *iff*, their durations are same, each of their renewable and non-renewable resource requirements are same or more in mode j_2, and each time period's associated *cash inflow* in mode j_1 is same or more than the corresponding cash inflow in mode j_2. Identical modes in these problem instances are defined by identical resource requirements, identical durations, and identical cash flows. In randomly generated problem instances such a situation rarely arises, and if found, only one may be included in consideration to save computational requirements.

We compute the earliest start time (EST) and latest start time (LST) for all activities, without considering the resource constraints using MPM (Metra Potential Method). We also determine the shortest mode of each activity (m_{is}) and its duration in this mode (p_{is}). LST underestimates the latest start time as when resource constraints are considered, it may have to be delayed further. The latest acceptable start time (LAST) for an activity is determined by subtracting the duration of its shortest mode and the length of the shortest path to completion of the project, from the project's due date (DD), i.e. DD – d_{is}, which represents an upper bound on the start time of activity a_i. Note that since all other modes of the activity are same or greater in duration, this upper bound is permissive, i.e. it may yield states where an activity completes after the due date, while it does not disqualify any states which may yield an optimal solution. To conserve repetitive computational effort, we also calculate and preserve the discounting factors for various time periods at the given rate of discount, β, for all time periods up to the due date. The time period just after the due date has a prohibitively large negative penalty associated with it, which enforces the completion of the project by the due date. As the NPV of a project may improve if we right shift an

activity (for example by delaying current activity's cash out flow, or to make resources available for another activity in such a mode which yields a higher cash inflow), hence, we consider the generation of all Delayed Resource Satisfying Sets (DRSS) for the finish time of the activity and its mode under consideration, for each time period, up to the due date. A DRSS is generated by right shifting an activity in its selected mode, one time period at a time. One DRSS is generated for each time period up to DD - d_{ij}, where d_{ij} is the duration of activity i in its mode j. This concept is explained with an example involving activity mode(s) with negative cash flows (after an example involving only positive cash flows) later in this paper.

Figure 1: Example 1 Project for Non-regular Objective (NPV)



Legend
Actv no., mode: duration, needed renewable resources R1 and R2, needed non-renewable resources NR1 and NR2.

The procedure for determining the start times of activities in a child state for the NPV objective is as follows. If an activity with a positive terminal cash flow in its given mode, a_{ij}, was in progress in the parent state, then its start time in the parent state is retained in the child state. If the activity was not present in the parent state, its start time is set as the current time (the decision point) in the child state.

The treatment for an activity and mode with a negative terminal cash flow is different. If the activity was in progress in the parent state and its start time is less than the decision point in child state, its start time is retained. However, if the activity's start time is greater than the decision point, then all possible start times from current time to the activity's

LAST in its current mode are considered and a corresponding child state generated. Only activity and modes with negative terminal cash flows are considered for right shifting as right shifting the activity and modes with positive terminal cash flows would reduce the NPV. Due to this, even a small NPV problem instance is more difficult to solve, and results in generation of a large number of states. The one child set rule, explained in Dayal and Verma (2014), used for regular measures is no longer applicable. Necessary alterations to the local left-shift rule and the dominance pruning rule are explained below.

The local left-shift rule for NPV objective involves shifting of activities with *positive terminal cash flows* only. An activity a_i in mode j with start time s_i in a partial schedule X is left-shiftable if $CF_i \geq 0$, and a_i can be started earlier, in the same mode j, without violation of precedence and resource restrictions, and without affecting the start time of any other activity in F_X or A_X (i.e. F_X U A_X). A DRSS (A_D) of state X, at a decision point dp_X, is left-shiftable if it contains an activity which is left-shiftable.

4.2. Left-shift (LS) Pruning Rule for Maximizing NPV

If a DRSS A_D at a decision point dp_X in state X is left-shiftable then do not generate any DRSS corresponding to this RSS. This is the local version of the left-shift rule. The global version of left-shift rule incorporates substantially more computational effort. To explain the LS rule, consider the problem instance in Example 1 shown in Figure 1 and its associated payment schedule in Table 1.

Table 1: Payments Associated with the Example 1 Problem

(1)	(2)	(3)	(4)	(5)	(6)
Actv	**Mode**	**Durn**	**At Start**	**During Activity's Progress**	**At end**
1	a	0	0	0	0
2	a	4	61	5,5,9,4	56
3	a	6	30	4,8,5,9,10,8	75
4	a	5	20	7,1,8,9,1	-4
5	a	8	4	1,8,1,10,3,8,7,1	-30
6	a	5	54	5,8,10,5,5,	98
6	b	7	77	9,8,7,6,6,9,7	55
7	a	0	0	0	0

Due date: 25. Bonuses are...Due date-3: 78, Due date-2: 63, Due date-1: 47, Due date: 31.

In the above problem, R1 and R2 represent the two types of renewable resources and NR1 and NR2 represent the two types of non-renewable resources. For each time period of each

activity's each mode, and at the start and end of the activity, there are associated payments, which are given in Table 1. The search tree states generated using Breadth-first NPV are shown in Table 2. The pruning of partial solutions (states) due to the left-shift rule is indicated by embedded comments. An explanation of the generation of states follows thereafter.

Table 2: Breadth-first NPV Search with only LS Rule in Project of Example 1

(1)	(2)	(3)	(4)	(5)		(6)	(7)	(8)
State	Level	Parent	Decision Time dp_x	Completed Actvs (F_x)	Value Fnsh	Actvs in Prog ($A_{x)}$	Value Prog	Feasible RSS
S1	0	--	0	{}	0	{a11}	0	{a21}{a21,a41} {a31}{a31,a41} {a41}
S2	0	S1	4	{1}	0	{a21}	137.23	{a31}{a31,a41} {a41}
S3	0	S1	4	{1}	0	{a21,a41}	178.7	
S4	0	S1	10	{1}	0	{a31}	111.28	
S5	0	S1	5	{1}	0	{a31,a41}	152.75	
S6	0	S1	5	{1}	0	{a41}	41.47	
S7	1	S2	14	{1,2}	137.23	{a31}	106.92	
S8[1]	1	S2	14	{1,2}	137.23	{a31,a41}	146.76	
S9[2]	1	S2	14	{1,2}	137.23	{a41}	39.84	

...and so on.

[1,2] Pruned as Left-shiftable

Note, activity a_{41} is simultaneously and resource feasibly processable with a_{21}, hence, it is left-shiftable. In actual implementation, the states S8 and S9 are not generated at all, and are shown here only for explanation. The progress of algorithm with both rules solving the complete problem instance follows later.

We begin the algorithm with starting state S1. The candidate activities are 1, 2, and 3. The resource feasible activity-mode combinations which can be processed are indicated in column 9. Each of these is developed into a partial schedule as state numbers S2, S3, S4, S5, and S6, respectively, and appended to the tree at Level 1 (which indicates that one activity is completed in these partial schedules, namely, the dummy starting activity 1). After completing the processing of all states at Level 0 we proceed to the next level. Column 4 indicates the earliest completion time of any activity in progress where the next decision is required to be taken, i.e. the decision point.

In breadth-first order, the state S2 is eligible for processing, where activity a_{11} is completed and a_{21} is in progress. The next set of feasible RSS is again indicated in column 9. Only the RSS $\{a_{31}\}$ is processed and state S7 is generated, as the other two RSS are left-shiftable (note only activity-modes with positive cash flows are considered for left-shift). The value of finished activity (its contribution to the project's NPV) is computed and indicated in column 6, while the expected additional value on completion of the activity (or activities) in progress is shown in column 8.

4.3. Dominance Pruning Rule

We now explain the implementation of the Dominance Pruning Rule for NPV objective and show its implementation using the same example problem instance, Example 1.

Dominance Pruning (DP) Rule for Maximizing NPV: If at any time during the execution of the algorithm there are two states X and Y, such that:

(i) $F_X = F_Y$, i.e., the activities completed (or finished sets) are same;

(ii) $A_X = A_Y$, i.e. activities in progress and their corresponding modes are same;

(iii) the residual of each non-renewable resource, after consumption by all activities completed or allocation to all activities in progress in set X, is same or more than in set Y;

(iv) the starting time in state X of each activity in A_X *with a non-negative terminal cash flow* is less than or equal to its starting time in state Y, i.e. $s_{ix} \leq s_{iy}$, $a_i \in A_Y$, $CF_i \geq 0$;

(v) the starting time in state X of each activity in A_X *with a negative terminal cash flow* is equal to its starting time in state Y, i.e. $s_{ix} = s_{iy}$, $a_i \in A_Y$, $CF_i < 0$;

(vi) the total expected cash flow from finished activities in state X is greater than or equal to that from state Y, i.e. $G_X \geq G_Y$, where the total expected cash flow from finished activities for a state I (G_I) is defined as, $G_I = \sum_{i=1}^{N} E_i$ for all $a_i \in F_I$, and E_i is the expected terminal cash flow from activity a_i;

then prune state Y from the search tree, as state X dominates state Y.

As an example, consider the same problem instance, Example 1, again. We shall now solve it applying only the Dominance Pruning rule. The generation of states is shown in

Table 3 and explanation follows thereafter.

Table 3: Breadth-first NPV Search with only DP Rule in Project of Example 1

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	
State	Level	Parent	Decision Time dp_x	Completed Actvs (F_x)	Value Fnsh	Actvs in Prog (A_x)	Value Prog	Feasible RSS
S1	0	--	0	{}	0	{a11}	0	{a21}{a21,a41} {a31}{a31,a41}{a41}
S2	0	S1	4	{1}	0	{a21}	137.23	{a31}{a31,a41}{a41}
S3	0	S1	4	{1}	0	{a21,a41}	178.7	{a31}{a31,a41}{a41}
S4	0	S1	10	{1}	0	{a31}	111.28	
S5	0	S1	5	{1}	0	{a31,a41}	152.75	
S6	0	S1	5	{1}	0	{a41}	41.47	
S7	1	S2	14	{1,2}	137.23	{a31}	106.92	
S8	1	S2	14	{1,2}	137.23	{a31,a41}	146.76	
S9	1	S2	14	{1,2}	137.23	{a41}	39.84	
S10[1]	1	S3	14	{1,2}	137.23	{a31}	106.92	
S11[2]	1	S3	5	{1,2}	137.23	{a31,a41}	148.39	
S12	1	S3	5	{1,2}	137.23	{a41}	41.47	

[1] Dominated by S7
[2] Dominates S8

Note, the LS rule has not been applied. State S7 dominates state S10, and state S11 dominates state S8. The progress of algorithm with both rules solving the complete problem instance follows later.

We begin the algorithm with starting state S1. The generation of states at first level is similar to the previous explanation. After the child states of (parent) state S2 have been generated, the next parent state, S3, is taken for processing. The feasible set of RSSs is: $\{a_{31}\}$, $\{a_{31},a_{41}\}$, and $\{a_{41}\}$. The state S10 is generated developing the RSS $\{a_{31}\}$ and DP rule applied. This state is pruned by the previously developed state S7. The next RSS, $\{a_{31},a_{41}\}$, is then developed and DP rule applied again. It is seen that state S11 dominates state S8, hence, state S8 is removed. Similarly, state S9 is dominated by state S12. After processing state S3 from level one completely, the processing continues to the next available state at level one, i.e. state S4, and so on. We present the complete solution to the example problem, Example 1, with both the rules applied, below.

The complete breadth-first algorithm with pruning rules for maximization of NPV is thus, as follows.

Algorithm Breadth-first NPV with Pruning Rules

Preprocessing	remove infeasible modes, redundant resources, and inferior modes
Preparing	calculate m_{is}, est_i, lst_i, $last_i$, and CF_i for each activity a_i in the project
Step 1 (Initialization)	create the root state I at level 0 in the search tree
Step 2 (Loop-1)	for level L from 0 to N-1 do
Step 3A (Loop-2)	for each state X at level L do
Step 3B (Expansion)	determine dp_X, completed activities, and child states' level (L_c) construct K_X and all the RSSs
Step 3C (LSR)	for each RSS (A), if the left-shift rule does not apply, construct DRSSs for each DRSS (A_D), construct a child state at level L_c
Step 3D (DPR)	apply the dominance pruning rule to all states at level L_c
Step 4 (Termination)	Traverse the states at level N and output the complete optimal schedule associated with the state X with maximum NPV

Following are the problem instance's states generated when the example problem instance is solved with both the rules applied.

Table 4: Breadth-first NPV Search with both Rules

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
State	Level	Parent	Decision Time dp_x	Completed Actvs (F_x)	Value Fnsh	Actvs in Prog (A_x)	Value Prog	Feasible RSS
S1	0	--	0	{}	0	{a11}	0	{a21} {a21,a41} {a31} {a31,a41} {a41}
S2	1	S1	4	{1}	0	{a21}	137.23	{a31} {a31,a41}* {a41}*
S3	1	S1	4	{1}	0	{a21,a41}	178.7	{a31}{a31,a41} {a41}
S4	1	S1	10	{1}	0	{a31}	111.28	{a21} {a21,a41}* {a41}*
S5	1	S1	5	{1}	0	{a31,a41}	152.75	{a21}*{a31}
S6	1	S1	5	{1}	0	{a41}	41.47	{a21}* {a31}*
S7	1	S2	14	{1,2}	137.23	{a31}	106.92	{a41}*
S8[1]	1	S3	14	{1,2}	137.23	{a31}	106.92	
S9	1	S3	5	{1,2}	137.23	{a31,a41}	148.39	{a31}
S10[2]	1	S3	5	{1,2}	137.23	{a41}	41.47	{a31}
S11	1	S4	14	{1,3}	111.28	{a21}	124.17	{a41}*
S12	1	S5	10	{1,4}	41.47	{a31}	111.28	{a21}
S13	2	S9	14	{1,2,4}	178.7	{a31}	106.92	{a51}{a51,a61} {a51,a62}{a61} {a62}
S14	2	S12	14	{1,3,4}	152.75	{a21}	124.17	{a51}{a51,a61} {a51,a62}{a61} {a62}
S15	3	S13	22	{1,2,3,4}	285.62	{a51}	11.78	{a61}* {a62}*
S16	3	S13	19	{1,2,3,4}	285.62	{a51,a61}	167.64	{a51}
S17	3	S13	21	{1,2,3,4}	285.62	{a51,a62}	166.79	{a51}
S18	3	S13	19	{1,2,3,4}	285.62	{a61}	155.86	{a51}*
S19	3	S13	21	{1,2,3,4}	285.62	{a62}	155.01	{a51}*
S20[3]	3	S14	22	{1,2,3,4}	276.92	{a51}	11.78	
S21[4]	3	S14	19	{1,2,3,4}	276.92	{a51,a61}	167.64	
S22[5]	3	S14	21	{1,2,3,4}	276.92	{a51,a62}	166.79	
S23[6]	3	S14	19	{1,2,3,4}	276.92	{a61}	155.86	
S24[6]	3	S14	21	{1,2,3,4}	276.92	{a62}	155.01	
S25	4	S16	22	{1,2,3,4,6}	441.48	{a51}	11.78	{a71}
S26[7]	4	S17	22	{1,2,3,4,6}	440.63	{a51}	11.78	
S27	5	S25	22	{1,2,3,4,5,6}	515.86	$\{a_{71}\}$	0.00	

* Pruned by Left Shift Rule
[1]Dominated by S7 [2] Dominated by S15,[3] Dominated by S16, [4] Dominated by S17, [5] Dominated by S18, [6] Dominated by S19, [7]Dominated by S25.

Example1 Due date:25 Optimal NPV makespan: 22 Optimal NPV: 515.855591

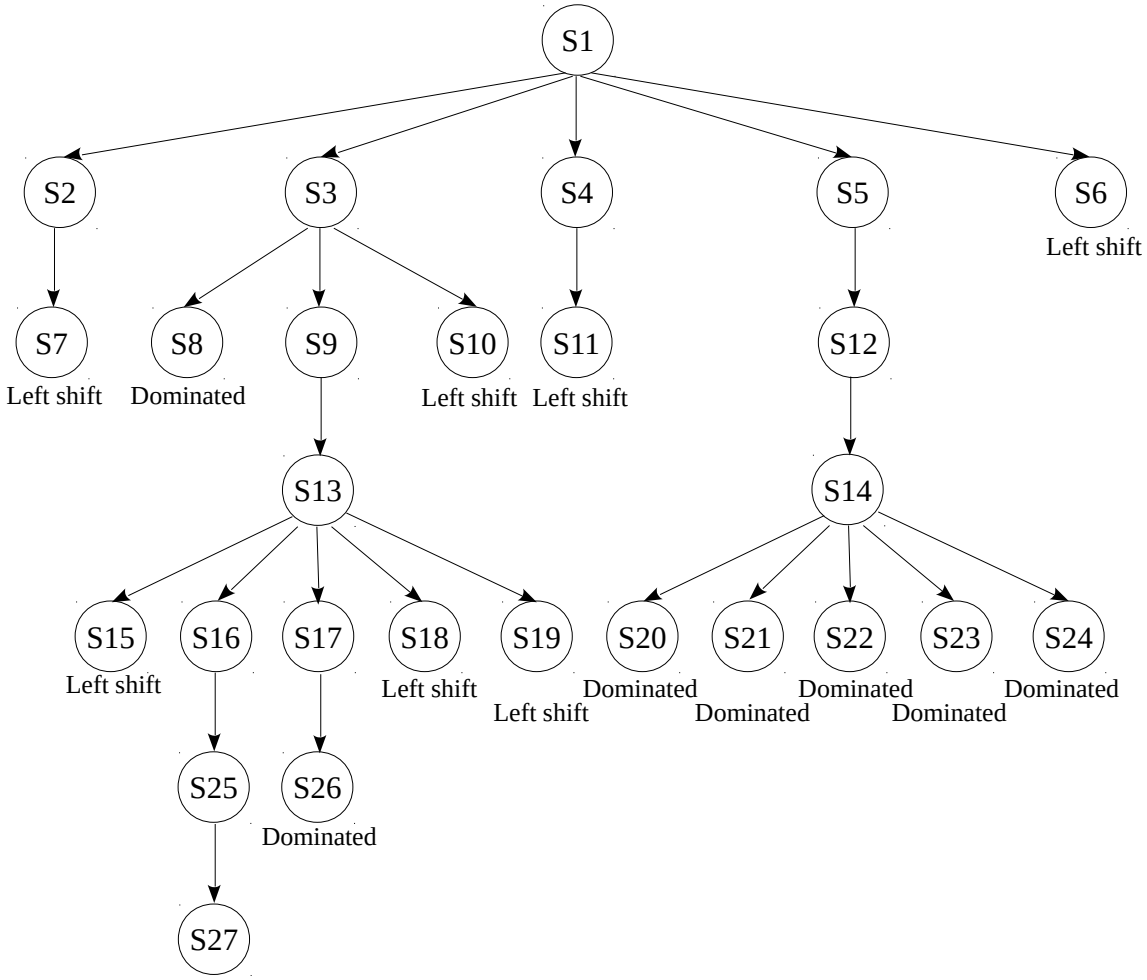
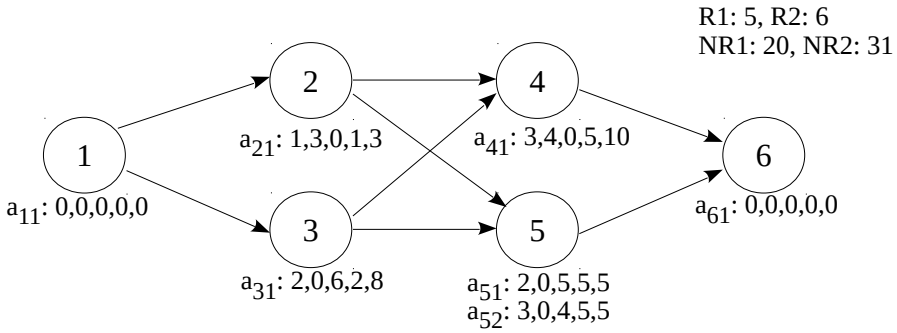


Figure 2: The Tree for Example 1 Project for Non-regular (NPV) Objective

We begin the algorithm with starting state S1. The candidate activities are 1, 2, and 3. The resource feasible activity-mode combinations which can be processed are indicated in column 9. Each of these is developed into a partial schedule as state numbers S2, S3, S4, S5, and S6, respectively, and appended to the tree at Level 1 (which indicates that one activity is completed in these partial schedules, namely, the dummy starting activity 1). After completing the processing of all states at Level 0 we proceed to the next level. Column 4 indicates the earliest completion time of any activity in progress where the next decision is required to be taken, i.e. the decision point.

In breadth-first order, the state S2 is eligible for processing, where activity a_{11} is completed

and a_{21} is in progress. The next set of feasible RSS is again indicated in column 9. Only the RSS $\{a_{31}\}$ is processed and state S7 is generated, as the other two RSS are left-shiftable (note only activity-modes with positive cash flows are considered for left-shift). These two RSS are marked with 'LS' in Table 4. The value of finished activity (its contribution to the project's NPV) is computed and indicated in column 6, while the expected additional value on completion of the activity (or activities) in progress is shown in column 8.



Legend
Actv no., mode: duration, needed renewable resources R1 and R2, needed non-renewable resources NR1 and NR2.

Figure 3: Example 2 Project for DRSS

The state S3 is processed next, where activity a_{21} completes and the next feasible RSS are $\{a_{31}\}$ with retraction of a_{41}, $\{a_{31},a_{41}\}$ and $\{a_{41}\}$. State S8, developed using the RSS $\{a_{31}\}$, is dominated by state S7. A child state corresponding to each of the remaining two RSS is generated and appended to the tree at appropriate level (level 2); these are states S9 and S10, respectively. The npv values due to the finished activities and activities in progress are computed and shown in respective columns 6 and 8.

State S4 follows, and from the three feasible RSSs for this state, two are left-shiftable and hence not developed (i.e. pruned). The state S11 is generated as a child state for S4 using the RSS $\{a_{21}\}$ and associated values computed.

State S5 is the next state to be processed as parent state, generating RSS $\{a_{21}\}$ which is left-shiftable and $\{a_{31}\}$ which is developed as state S12 and appended at level 2. This completes the processing of all states at level 1, and the states at level 2 are now taken up for processing.

Among the five states available for processing at level two, the RSS for three are left-

shiftable. Thus, only two states, S9 and S12, result into child states, S13 and S14, respectively, for level 3.

Processing S13 first, five RSSs are generated and a child state for each, S15 to S19, is developed and appended at level 4. Note that one child rule is not applicable for non-regular measures (if applied, it may result into a sub optimal solution). Thus, five child states from parent state S13 are generated. Five child states would also be generated from the parent state S14, however, these are dominated by the child states of S13, and hence, pruned.

Processing of the next level (level four) is now started and the child states of S13 are processed. The RSS generated for S15, S18 and S19 are left-shiftable, hence, no states are generated for these. The child state from S16, namely S25, is generated and appended to the search tree at level 5. It dominates the state S26 resulting from the processing of state S17. Finally, we schedule the 'bonus' dummy activity to add the applicable bonus to value obtained at finish of S25 which gives the optimal NPV and applicable makespan. The mutual relationships of the states are pictorially represented in the Figure 2.

We now provide a small example, Example 2, involving negative cash flows and explain the implementation of DRSS. Consider the example problem instance shown in Figure 3. The payments associated with the activities in their given modes and the bonus/penalty for completion within due date for the above problem instance are as given in Table 5.

Table 5: Cash Flows associated with the Example 2

(1)	(2)	(3)	(4)	(5)	(6)
Actv	**Mode**	**Duration**	**Cash Outflow At Start**	**Cash Outflow For all t>0**	**Cash Outflow At end**
1	a	0	0	0	0
2	a	1	5	1	5
3	a	2	5	1,2	5
4	a	3	5	1,2,3	5
5	a	2	**-5**	1,2	2
5	b	3	5	1,2,3	2
6	a	0	0	0	0
Due date: 6. Bonuses are: Three days before Due date: 9, Two days before Due date: 2, One day before Due date: 7, On Due date: 8.					

The generation and pruning of states in breadth-first NPV algorithm is given in Table 6 below and explained thereafter.

Table 6: Breadth-first NPV Search in Example 2 Project for DRSS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
State	Level	Parent	Decision Time dp_x	Completed Actvs (F_x)	Value Fnsh	Actvs in Prog ($A_{x)}$	Value Prog	Feasible RSS
S1	0	--	0	{}	0	{a11}	0	{a21}{a21,a31} {a31}
S2*	1	S1	1	{1}	0	{a21}	10.94	{a31}
S3	1	S1	1	{1}	0	{a21,a31}	23.79	{a31}
S4*	1	S1	2	{1}	0	{a31}	12.85	{a21}
S5	2	S3	2	{1,2}	10.94	{a31}	12.85	{a41}{a41,a51} {a41,a52}{a51} {a52}
S6*	3	S5	5	{1,2,3}	23.79	{a41}	15.4	{a51}{a52}
S7	3	S5	4	{1,2,3}	23.79	{a41,a51}	15.32	{a41}
S8	3	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	{a61}
S9	3	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	{a51}{a52}
S10	3	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	{a41}{a41,a51} {a51}
S11	3	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	{a41}{a41,a51}
S12	3	S5	5	{1,2,3}	23.79	{a41,a52}	27.95	{a41}
S13	3	S5	4	{1,2,3}	23.79	{a51}	-0.09	(LS)
S14	3	S5	5	{1,2,3}	23.79	{a51}	-0.09	(LS)
S15	3	S5	6	{1,2,3}	23.79	{a51}	-0.09	(LS)
S16	3	S5	7	{1,2,3}	23.79	{a51}	-0.08	(LS)
S17	3	S5	8	{1,2,3}	23.79	{a51}	-0.08	(LS)
S18	3	S5	5	{1,2,3}	23.79	{a52}	12.55	(LS)
S19	3	S6	7	{1,2,3,4}	39.19	{a51}	-0.08	{a61}
S20	3	S6	8	{1,2,3,4}	39.19	{a51}	-0.08	{a61}
S21	3	S7	5	{1,2,3,5}	23.7	{a41}	15.4	{a61}
S22#	3	S8	5	{1,2,3,4,5}	31.5	{a61}	0	{a61}
S23	3	S9	6	{1,2,3,4}	39.19	{a51}	-0.09	{a61}
S24	3	S9	8	{1,2,3,4}	39.19	{a52}	12.18	{a61}
S25#	3	S10	7	{1,2,3,4}	39.19	{a51}	-0.08	{a61}
S26#	3	S10	8	{1,2,3,4}	39.19	{a51}	-0.08	{a61}
S27#	3	S10	8	{1,2,3,4}	39.19	{a51}	12.18	{a61}
S28#	3	S11	8	{1,2,3,4}	39.19	{a51}	-0.08	{a61}
S29#	3	S11	8	{1,2,3,4}	39.19	{a51}	12.18	{a61}
S30#	3	S12	5	{1,2,3,4,5}	44.13	{a61}	0	{a61}
S31#	4	S19	7	{1,2,3,4,5}	39.11	{a61}	0	--
S32#	4	S20	8	{1,2,3,4,5}	39.11	{a61}	0	--
S33#	4	S23	6	{1,2,3,4,5}	39.11	{a61}	0	--
S34#	4	S24	8	{1,2,3,4,5}	51.37	{a61}	0	--
S35#	4	S21	5	{1,2,3,4,5}	31.5	{a61}	0	--
S36#	5	S34	8	{1,2,3,4,5,6}	51.37	--	0	

*Left-shift, # DP by S30

Optimal NPV value: 51.37, Optimal NPV makespan: 8

We begin the algorithm with (dummy) activity one in progress in state S1. The resource feasible sets to proceed with are $\{a_{21}\}$, $\{a_{21},a_{31}\}$, and $\{a_{31}\}$. These are developed into states at the level one S2, S3, and S4. The states S2 (to be continued with a_{31} on completion of a_{21}) and S4 (to be continued with a_{21} on completion of a_{31}) are pruned by the left-shift rule. The states S3 is developed into state S5 at the next level. The NPV of activities completed and activities in progress are shown in columns 6 and 8 respectively.

As the state S5 is developed, an activity $\{a_{51}\}$ is encountered that has a net negative cash flow. For maximization of NPV we would like to delay the activities with negative cash flows. However, the given due date keeps a cap on the maximum right shift for such an activity. Whenever such an activity-mode combination is encountered, we develop and generate one partial schedule each for it (in that mode) starting at all time periods from current time, such that it completes before or on the due date. Note that if it has successors, then we keep allowance for the shortest path from its completion in that mode to the end of the project. As already stated earlier, the shortest paths for all activities are computed by the established Metra Potential Method (MPM) ignoring the resource constraints. All the child states generated represent the complete set of delayed resource satisfying sets (DRSS) for the activity with negative flow encountered. If two or more activities with such modes whose final cash flows are negative are encountered, the tree's size explodes! This gives rise to a large number of states due to which the NPV objective (and in general, most of the non-regular objective problems) become extremely difficult to solve exactly using tree search procedures.

For the state S5, with the RSS $\{a_{41},a_{51}\}$ and $\{a_{51}\}$, both of which involve negative cash flows due to the activity $\{a_{51}\}$, five child states each are generated considering the time of its start, such that it completes on or before the due date. These states are, respectively, S7 to S11 for $\{a_{41},a_{51}\}$ and states S13 to S17 for $\{a_{51}\}$. A few of these states are pruned due to application of the NPV versions of the left-shift rule or the dominance pruning rule. The other states are developed one by one in breadth-first order. The pruning rule applied and the state causing the pruning are mentioned in the remarks column in Table 6.

It is extremely rare in NPV problems to find multiple optimal solutions, and hence, the possibility of finding exact multi objective solutions, as in breadth-first regular measure

algorithm, appears to be negligible.

We now describe our single-processor best-first algorithm for non-regular performance measures, namely the NPV.

4.4. Single-processor best-first NPV algorithm

The Best-first NPV algorithm differs from Breadth-first NPV in the manner of selection of the next state to process. The criteria for selection of the next state to be processed are the best estimated project NPV ignoring the resource constraints. The partial schedule is converted into a pseudo-complete schedule and its project NPV is estimated as the sum of NPV of: (a) the activities in F_X, (b) the activities in A_X, (c) the activities not yet scheduled by determining their best mode at dp_X, and assuming they all started at dp_X, and (d) the applicable project bonus as per the latest finish time determined from all the incomplete and unscheduled activities. A tie in the forward estimates of two partial schedules is broken by considering the number of activities completed (the finished activities set, F_X) and choosing the larger; and further, by the smaller finish time (dp_X) of the partial schedules. Here, as the heuristic value over estimates the actual NPV, the first solution state selected yields a schedule of maximum NPV. Though two different schedules with same project NPV are theoretically possible, such cases are rare in practice. The optimal schedules revealed by Breadth-first and Best-first in our test sets are identical.

We present two example problems for the non-regular objective, NPV, below. The first considers only positive cash flows, while the next example also considers negative cash flows. For convenience, the example problem is the same as used in demonstrating the Breadth-first NPV algorithm above, i.e. in Example 1. We present the states generated by the Best-first NPV algorithm incorporating all pruning rules with the optimal solution obtained in Table 7.

Table 7: Best-first Search in Example 1 Project with Pruning Rules

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
State	Parent	Decision Time dp_x	Completed Actvs (F_x)	Value Fnsh	Actvs in Prog ($A_{x)}$	Value Prog	Upper Bound	Number Finshd	Res. Fsble RSS
S1	--	0	--	0	{1a}	0	0	0	{a21} {a21,a41} {a31}{a31,a41} {a41}
S2	S1	4	{1}	0	{a21}	137.23	531.86	0	{a31} {a31,a41} -LS{a41}-LS
S3	S1	4	{1}	0	{a21,a41}	178.7	533.49	0	{a31} {a31,a41} {a41}
S4	S1	10	{1}	0	{a31}	111.28	510.05	0	
S5	S1	5	{1}	0	{a31,a41}	152.75	529.31	0	{a21} -DP{a31}
S6	S1	5	{1}	0	{a41}	41.47	523.89	0	
S7	S3	14	{1,2}	137.23	{a31}	106.92	510.44	1	
S8	S3	5	{1,2}	137.23	{a31,a41}	148.39	531.64	1	{a31}
S9	S3	5	{1,2}	137.23	{a41}	41.47	530.58	1	
S10	S2	14	{1,2}	137.23	{a31}	106.92	510.44	1	
S11	S8	14	{1,2,4}	178.7	{a31}	106.92	515.86	2	{a51} {a51,a61} {a51,a62} {a61}{a62}
S12	S5	10	{1,4}	41.47	{a31}	111.28	514	1	
S13	S11	22	{1,2,3,4}	285.62	{a51}	11.78	503.87	3	
S14	S11	19	{1,2,3,4}	285.62	{a51,a61}	167.64	515.86	3	{a51}
S15	S11	21	{1,2,3,4}	285.62	{a51,a62}	166.79	515.01	3	
S16	S11	19	{1,2,3,4}	285.62	{a61}	155.86	515.28	3	
S17	S11	21	{1,2,3,4}	285.62	{a62}	155.01	514.21	3	
S18	S14	22	{1,2,3,4,6}	441.48	{a51}	11.78	515.86	4	{a71}
S19	S18	22	{1,2,3,4,5,6}	515.86	{a71}	0	515.86	5	

Example1 Due date:25 Optimal NPV makespan: 22 Optimal NPV value: 515.855591

The algorithm starts with the starting state S1 where only the dummy start activity a_{11} is scheduled. The next set of RSS is generated and each is developed into a state representing a partial solution. The RSS are: $\{a_{21}\}$, $\{a_{21},a_{41}\}$, $\{a_{31}\}$, $\{a_{31},a_{41}\}$, and $\{a_{41}\}$. These are respectively developed into the states S2, S3, S4, S5, and S6. The cash flow values of finished activities and activities in progress appear in columns 5 and 7 respectively, and the activities in progress in a state in column 6.

Note that for a makespan optimization, which requires **minimization**, the Best-first monotone heuristic underestimates the objective value, and the most promising state, which appears at the top of the heap, is processed first. However, for the NPV **maximization** objective, the upper bound over-estimates the possible net present value of the likely solution to be arrived at from a partial solution, and finally converges on the optimal value.

The upper bound for each of the new child states is computed and each is added to the heap. The most promising state, which appears at the top of the heap is then selected for development further, which in this case is the state S3, with a likely NPV value of 533.49. Its RSS are generated and developed into child states, namely, S7, S8 and S9. Their estimates of NPV too are computed and these are accordingly added to the heap, breaking any ties with the larger number of activities completed, and further by the lesser finish time.

The next state at the top of the heap is now processed, which is state S2. Its RSS are $\{a_{31}\}$, $\{a_{31},a_{41}\}$, and $\{a_{41}\}$. Only $\{a_{31}\}$ is developed into a state, i.e. state S10, as the other two are pruned by the left-shift rule, as they involve activities in modes with positive cash flows which can resource-feasibly be left-shifted. The new child state is also added to the heap at the right place.

Taking the next state from the top of the heap, state S8, its RSS is generated, which is only $\{a_{31}\}$. This is developed into the child state of S8, i.e. state S11. Its forward estimate of NPV value is computed and the new state S11 is added to the heap at its right place.
State S5, which is now at the top of the heap, is taken for expansion. The RSS generated are: $\{a_{21}\}$ and $\{a_{31}\}$. The state corresponding to $\{a_{21}\}$ is dominated and hence pruned, while the child state generated using the RSS $\{a_{31}\}$, i.e. state S12, is appended to the heap at its right place.

The state S11 now appears at the top of the heap, whose RSS are now generated. These are: $\{a_{51}\}$, $\{a_{51},a_{61}\}$, $\{a_{51},a_{62}\}$, $\{a_{61}\}$, and $\{a_{62}\}$. These are developed into the states S13, S14, S15, S16, and S17. Their upper bounds are computed and each is appended to the heap at the right place. State S14 appears at the top of the heap now and is taken for development

further. Its child state S18 is generated which also reaches the top of the heap when appended to it. It is further developed into state S19 yielding the optimal solution. The optimal NPV for the project is 515.85591 and the makespan for this schedule is 22.

Next we consider another example problem instance with a mode of an activity with negative cash flows to demonstrate its processing by the algorithm. The example taken is the same as for Breadth-first NPV objective earlier, i.e. in Example 2. The development of states for this problem instance is as shown in Table 8.

Table 8: Best-first Search in Example 2 Project with Pruning Rules

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
State	Parent State	Decision Time dp_x	Completed Actvs (F_x)	Value Fnsh	Actvs in Prog (A_x)	Value Prog	Upper Bound	Number Finshd	Res. Fsble RSS
S1	--	0	--	0	{1a}	0	0	0	{a21}{a21,a31} {a31}
S2	S1	1	{1}	0	{a21}	10.94	60.72	0	
S3	S1	1	{1}	0	{a21,a31}	23.79	60.85	0	{a31}
S4	S1	2	{1}	0	{a31}	12.85	60.35	0	
S5	S3	2	{1,2}	10.94	{a31}	12.85	60.57	1	{a41}{a41,a51} {a41,a52}{a51} {a52}
S6	S5	5	{1,2,3}	23.79	{a41}	15.4	60.19	2	{a51}{a52}-LS
S7	S5	4	{1,2,3}	23.79	{a41,a51}	15.32	47.93	2	
S8	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	47.93	2	
S9	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	47.93	2	
S10	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	47.93	2	
S11	S5	5	{1,2,3}	23.79	{a41,a51}	15.32	47.93	2	{a51}{a52}
S12	S5	5	{1,2,3}	23.79	{a41,a52}	27.95	60.57	2	{a61}
S13	S5	4	{1,2,3}	23.79	{a51}	-0.09	47.62	2	
S14	S5	5	{1,2,3}	23.79	{a51}	-0.09	47.47	2	
S15	S5	6	{1,2,3}	23.79	{a51}	-0.09	47.33	2	
S16	S5	7	{1,2,3}	23.79	{a51}	-0.08	47.18	2	
S17	S5	8	{1,2,3}	23.79	{a51}	-0.08	47.04	2	
S18	S5	5	{1,2,3}	23.79	{a52}	12.55	60.11	2	
S19	S12	5	{1,2,3,4,5}	44.13	{a61}	0	4.13	3	
S20	S6	7	{1,2,3,4}	39.19	{a51}	-0.08	47.93	3	
S21	S6	8	{1,2,3,4}	39.19	{a51}	-0.08	47.93	3	{a61}
S22	S21	8	{1,2,3,4,5}	39.11	{a61}	0	39.11	4	
S23	S11	8	{1,2,3,4}	39.19	{a51}	-0.08	47.93	3	
S24	S11	8	{1,2,3,4}	39.19	{a52}	12.18	60.19	3	{a61}
S25	S24	8	{1,2,3,4,5}	51.37	{a61}	0	51.37	4	

Optimal NPV: 51.372875, Optimal NPV makespan: 8

The algorithm begins with starting state S1 in which dummy activity a_{11} is scheduled. The next set of RSS are generated, which are: $\{a_{21}\}$, $\{a_{21},a_{31}\}$, and $\{a_{31}\}$. The corresponding partial solutions, i.e. states S2, S3, and S4 are generated and added to the heap. The state at the top of the heap is next taken for development, i.e. state S3. Its RSS are generated, i.e. only $\{a_{31}\}$, and the child state developed along with its upper bound and added to the heap at the right place.

State S5 now appears at the top of the heap with the best estimated NPV, and hence, is processed next. Its RSS are generated, which are: $\{a_{41}\}$, $\{a_{41},a_{51}\}$, $\{a_{41},a_{52}\}$, $\{a_{51}\}$, $\{a_{52}\}$. Note that activity a_{51} has a negative cash flow. This activity and mode combination appears in two RSSs. For both of these RSSs, a child state is generated for all its feasible start times from the maximum of previous finish time and the activity's earliest start time, and up to the latest start time of the activity. This gives rise to a large number of child states to be appended to the heap, and demonstrates the difficulty of the problem in being solved. Upper bounds of all the child states are computed and they are appended to the right place in the heap.

The next partial solution state from the top of the heap is next selected for development, which is S12. Only the last dummy activity remains to be scheduled in this state, which is scheduled and the upper bound of NPV computed. The new child state is then added to the heap at the right place. The next state from the top of the heap, i.e. state S6, is then selected for processing. Its RSSs are generated and each developed into a partial solution state (i.e. states S20 and S21) and appended to the heap. From the top of the heap again, another state, state S21 now, is selected for development and its child state S22 generated and appended to the heap. The state S11 now appears at the top of the heap, which is developed next, generating the child states S23 and S24. State S24 appends at the top of the heap and is therefore selected next for development, generating the state S25 with the optimal NPV value:$\sum_{i=1}^{N} f_i - b_i$.

In larger problems facing several activities with multiple modes yielding a net negative cash flow, the complexity of the problem rises many folds, and so does the time taken to solve them.

5. Theoretical results

We now present proof of the optimality of the solutions generated by the two algorithms. We use the following abbreviations to represent versions of the algorithm: (a) BRD-NPV and BST-NPV, the Breadth-first NPV and Best-first NPV algorithms without any pruning rules; (b) BRD-NPV-ALL and BST-NPV-ALL, hypothetical versions of the two NPV algorithms with right-shift permitted for activities *with positive or negative terminal cash flows*; (c) BRD-NPV-DP for the algorithm with dominance pruning rule; (d) BRD-NPV-LS for the algorithm with left-shift pruning rule; and (e) BRD-NPV-DPLS for the algorithm with both pruning rules. We wish to establish that both algorithms, with pruning rules, generate optimal solution to the maximization of NPV problem.

Theorem 1: BRD-NPV-ALL generates *all* possible feasible complete schedules.
Proof: Since the method of generation of states is exhaustive, and activities are scheduled at all possible times consistent with the precedence, renewable, and non-renewable resource constraints without pruning any states, the result immediately follows. Note that as activities may be retracted, some left-shiftable schedules are likely to be additionally generated. □

Corollary 1: BRD-NPV-ALL generates an optimal schedule, if one exists.
Proof: Assume that an optimal schedule exists, then by Theorem 3.1, as all possible schedules are generated; BRD-NPV-ALL yields an optimal schedule. It is noteworthy that if a feasible schedule exists, an optimal schedule also exists. □

We avoid the generation of problems with infeasible due dates when we use our payment schedule generator for the PSPLIB problems. Now consider the adoption of the dominance pruning rule to BRD-NPV-ALL.

Lemma 1: If a state X is pruned by another state Y during the execution of BRD-NPV-DP, and X is optimizable, then so is Y.

Proof: Let state X, dominated by state Y during the execution of BRD-NPV-DP, be optimizable. Since state Y dominates state X, $F_Y = F_X$, and $A_Y = A_X$. Activities with positive terminal flow in state Y start respectively in same mode and no later than the corresponding activities in state X, and activities with negative terminal flow start respectively in same mode and at exactly the same time as in A_X. Both of the states X and

Y would be generated by BRD-NPV-ALL. Since X is optimizable, it has a successor state X' which is optimal. This state is also generated by BRD-NPV-ALL.

Consider the decision points t_1, t_2,..., and corresponding RSSs along the path in the search tree BRD-NPV-ALL from X to X'. Now, starting from state Y instead, and choosing exactly the same DMRSs at the same time instants t_1, t_2,..., obtain the schedule Y', which has no precedence or resource conflicts. Y' may have intervals of time where no activity is in progress, that is, it may have left-shiftable activities, each scheduled in one of its modes (note a left-shiftable activity in its given mode must have a positive terminal cash flow). Take each such activity and, in order of activity numbers, shift each as far left as possible retaining its mode, and without introducing any precedence or resource conflicts. Left-shifting an activity with a positive terminal cash flow can not decrease the NPV. Let the resulting schedule be Y". Y" can never have an NPV less than X', therefore Y" is optimizable and will be generated by BRD-NPV-DP as a successor of Y. Hence, Y is optimizable. □

Theorem 2: BRD-NPV-DP generates an optimal schedule.

Proof: This follows immediately from Lemma 3.1, as regardless of the states that are pruned, the search tree of BRD-NPV-DP will always contain a schedule which is optimal. □

We now prove that the introduction of the (new form of) left-shift rule for NPV maximization, i.e. BRD-NPV-DPLS, produces an optimal schedule.

Theorem 3: BRD-NPV-DPLS generates an optimal schedule.

Proof: By Theorem 3.2, it should suffice to prove that the schedule generated by BRD-NPV-DPLS is at least as good as that yielded by BRD-NPV-DP.

Note that we consider child states corresponding to all RSSs, and all the resource feasible MRSs are a subset of these RSSs. Consider a partial schedule, parent state X, whose child states are under development. There is a set of activities which is completed at or before the decision point dp_X, the earliest finish time of an activity in progress in the parent state X. Let the candidate set of activities for child states of parent X be represented by C. C includes the activities in progress in X but not completed at dp_X.

As stated earlier, in generating the child states, all renewable and non-renewable resource feasible sets (the RSSs) are considered for the parent state X. Let one child state generated, state Y, be such that in it an activity, a_i in mode j, is left-shiftable, i.e., it can be started at a time before dp_X, without violating any resource and precedence constraints and without affecting the start time of any other activity in progress. Let time t denote the earliest such time when activity a_i in mode j can be scheduled. Since a_i is ready at time t, it belongs to some (at least one) RSS at t in its resource feasible mode j, and thus, is included in at least one RSS, for which a child state has been generated (say, child state Z). As the terminal cash flow of a_i in mode j is non-negative, if an optimal schedule is generated by a child state of parent X scheduling activity a_i in mode j, at $t \geq dp_X$, then a schedule with at least the same NPV shall also be generated by state Z. Hence, state Y may be pruned without the loss of optimality. □

Theorem 4: If BRD-NPV-ALL generates an optimal schedule, BRD-NPV-DPLS will also generate an optimal schedule.

Proof: BRD-NPV-ALL generates DMRSs by assigning all possible start times to all activities in each RSS. Further, for an RSS, BRD-NPV-DP will right shift only the activities with negative terminal cash flows, in their respective modes.

Note that we consider child states corresponding to all RSSs, and all the resource feasible MRSs are a subset of these RSSs. Consider a partial schedule, parent state X, whose child states are under development. Let Y be an RSS, and let Y_1, Y_2, Y_3,..., Y_N, be the corresponding DRSSs. Let Y_i and Y_i' be two DRSSs such that the start times of activities with negative terminal cash flows are the same, and in Y_i the start times of activities with positive terminal cash flows are less than or equal to the corresponding start times of these activities, in same modes as in Y_i'. Since expected cash flow ($\sum_{i=1}^{N} E_i$) of finished activities is the same, Y_i' will be dominated and pruned by Y_i. For any state of the type Y_i', a state of the type Y_i will always be generated. BRD-NPV-DPLS will not generate Y_i', while it will generate and process Y_i, thus not missing the optimal solution. Therefore, BRD-NPV-DPLS will yield an optimal solution. The optimality of BST-NPV with both pruning rules can be proved similarly. □

6. Experimental observations

The computational machines used in the experiments are as follows.

Computational Machines: The development and initial experiments were carried out on a desktop machine (details provided below), while the tests were carried out using one CPU on a node in a high performance compute-cluster (HPCC) at the Physical Research Laboratory, Ahmedabad. The HPC has twenty-one nodes, each node with sixteen CPUs. Each compute-node is a collection of four boards, each with four Quad-Core AMD® Opteron™ Processor 8360 SE with 2511.578 MHz clock speed. At each node a total of 64 GB shared DDR2 SDRAM, 677 MHz is available, though in practice far less is used. The size of L1 cache is 128 KB, L2 cache is 512 KB, and L3 cache is 2048 KB. The core speed is 2500 MHz, integrated memory controller speed is 2000 MHz, and the system bus speed is 1000 MHz. The operating system used on the access server (the head node) is Red Hat Enterprise Linux 5 (RHEL 5), while on the compute nodes its light weight variant (or thin version without GUI for computational purposes) is deployed. All the algorithms were coded in C and compiled using Intel C Compiler without using any compile time parallelization or optimization directives. The details of computational machines used are as follows.

(A) Details of developmental and experimental operating systems used:

Operating System : (a) Fedora 11, 12, and 13 for development and testing.

: (b) RHEL 5 for experiments.

Compiler : GNU C Compiler (gcc), Intel C Compiler (icc).

Analysis tools : valgrind, Kcachegrind.

(B) Details of the developmental and testing platform (desktop) used:

Vendor_id : Intel Corp.

Model name : Intel® Pentium® D CPU 3.00GHz

CPU cores : 2

Core Speed (MHz) : 3000 (Max 4000 MHz)

L1 Cache Size (KB) : 2 x 16

L2 Cache Size (KB) : 2 x 1024

L3 Cache Size (KB) : Not provided.

System Bus Speed (MHz) : 533

(C) Details of compute-cluster platform used:

The computational experiments were conducted on a compute-cluster with sixteen processors, organized as four Quad-Core boards using AMD processors. The essential details are as given below.

Vendor_id : AuthenticAMD.

Model name : Quad-Core AMD® Opteron™ Processor 8360 SE

CPU cores : 4

Core Speed (MHz) : 2500

L1 Cache Size (KB) : 128

L2 Cache Size (KB) : 512

L3 Cache Size (KB) : 2048

System Bus Speed (MHz) : 1000

CMOS : 65nm SOI

IM Controller Speed (MHz) : 2000

Virtualization : Yes

Siblings : 4

Bogomips : 5026.04

TLB size : 1024 4K pages.

Address sizes : 48 bits physical, 48 bits virtual.

We discuss the results of our experiments with both the algorithms below.

Problem sets generated: Standard test problem sets for regular measures with known optimal results are available online, for example on websites of PSPLIB and ORLib. These problem sets have proved extremely useful as a benchmark for researchers in testing their new algorithms and sharing the obtained results. However, libraries for such problem sets for non-regular measures are missing and hence, newly generated test problem sets, typically a small number, have been used by most researchers. The non-regular objectives have attracted less attention due to the large computational requirements involved making even small problems hardly feasible within a limited time using a single modern processor. Multi-processor algorithms could prove beneficial for such problems.

In our research, we use PSPLIB problem sets with known optimal solutions as base problems, and generate random payment schedules using our own payment schedule instance generator. We consider payments at the start, the end, and for each unit time period of the duration of an activity, in each of its modes. Additionally, we consider a bonus payment for completion of the whole project within due date for four unit time periods. This model represents a generalized NPV problem model that covers most situations of interest. For each problem in a PSPLIB problem set, we generate ten random payment schedules. As the exact optimal makespan should be known before payments schedules can be generated in our NPV problem generator, the NPV versions for PSPLIB set j30 were not generated.

Computational environment: The algorithms were tested on a single processor in a cluster with Quad-Core AMD® Opteron® Processor 8360 SE, 2511.578 MHz running a thin (non-GUI) version of Red Hat Enterprise Linux 5.

Breadth-first NPV and Best-first NPV results

Both the algorithms yield an optimal solution on termination. In the problem sets where all the problems are solved by both the algorithms, Best-first NPV solves the problems much faster. A mutual comparison between these two algorithms reflects that while for regular measures Best-first is several times faster than Breadth-first; *same is not the case for non-regular measures*. Hence, the multi-processor implementation of Breadth-first NPV may yield substantially better results. The non-regular measures appear far more difficult to pursue, and both the algorithms are able to solve only small problems given a limited time. The results are briefly portrayed in Table 9 given below.

Table 9: Summary of Results of Computational Experiments

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
	% solved	Total time(s)	Mean time(s)	Std dev	Max time(s)	Solved <5 min	Solved <10 min	Solved <15 min	Solved <30 min
Problem set: PSPLIB j10 set with payment schedules for activities generated by our problem generator (total 5360)									
BRDMM	100%	7163	1.3	21.8	1582.5	5359	0	0	1
BSTMM	100%	5293	1	6.18	401.6	5359	1	--	--
Problem set: PSPLIB j12 set with payment schedules for activities generated by our problem generator (total 5470)									
BRDMM	100%	24116	4.4	22.4	1225	5465	4	0	1
BSTMM	100%	21694	4	14.4	655.3	5469	0	1	--
Problem set: PSPLIB j14 set with payment schedules for activities generated by our problem generator (total 5510)									
BRDMM	99.46%	380276	69.4	162.2	1620	5162	188	78	52
BSTMM	99.71%	332842	60.6	136.8	1734	5239	178	42	35
Problem set: PSPLIB j16 set with payment schedules for activities generated by our problem generator (total 5500)									
BRDMM	14.42%	314419	396.5	455.8	1787	469	126	79	119
BSTMM	19.60%	366570	340.1	402.4	1779	702	156	93	127
Problem set: PSPLIB j18 set with payment schedules for activities generated by our problem generator (total 552)									
BRDMM	3.08%	118644	697.9	533.5	1781	58	25	18	69
BSTMM	3.30%	133159	731.6	526.8	1759	49	38	29	66
Problem set: PSPLIB j20 set with payment schedules for activities generated by our problem generator (total 554)									
BRDMM	0%	--	--	--	--	0	0	0	0
BSTMM	0.18%	9876	987.6	580.6	1717	2	2	0	6

Note: To generate payment schedules with bonus for PSPLIB problem sets, the optimal makespan should be known. Hence, for PSPLIB set j30, only some of whose optimal solutions are known (solved by our Breadth-first and Best-first in limited time), the npv problems were not created.

Breadth first is unable to solve any problem from the npv problems generated for the PSPLIB problem set j20 within 30 minutes. While all the problems in the smaller sets j10, j12 are solved rapidly by both algorithms, further increase in problem size seems to handicap their ability substantially. The memory requirements by both the algorithms for small problems are modest. Nearly all problems are solved by both the algorithms in the NPV set j14 problems within thirty minutes. However, beyond this set only a limited numbers are solved in a run time limit of thirty minutes per problem.

For non-regular measure (npv), the Best-first algorithm appears to be not as superior to Breadth-first algorithm, as is the case in algorithms for regular measure (makespan). Problems involving negative net cash flows in an activity's mode appear to cause the size of the search tree to expand substantially. This is so because the activity mode combination with a negative net cash flow has to be tested at each time instant from a child state's dp_X to the activity-mode's latest acceptable schedule time, and this results in the generation of far too many states. If several activities with such cash flows are involved, the numbers of child states and the time needed to solve the problem rises substantially. Note that the number of problems, which were solved by both the algorithms (within thirty minutes of run time limit) becomes smaller as the problem size increases.

Best-first appears to perform better than the Breadth-first algorithm in npv problem sets generated from PSPLIB sets j14 and j16. As observed for set j18 the Best-first algorithm appears to need more time to solve the problems, however, this result is over only one hundred and twenty nine (129) problems solved by both algorithms within a limit of thirty minutes, out of a total of five thousand five hundred and forty (5540) problems.

6.1. Non-regular measure multi-objective solutions

As is expected, the advantage of multiple single objective optimal solutions and subsequent multi objective optimization, which is possible in regular measures as objectives, is not seen in non-regular measures. It appears that only tailor made special instances may possess multi objective optimization characteristics for non-regular measures. However, multiple feasible solutions (not all optimal) are indeed generated by Breadth-first NPV at the last level and these solutions consume varying amounts of non-renewable resources, too. Given the per unit costs of non-renewable resources, *a better solution which sacrifices the optimal NPV value but in return, preserves non-renewable resources*, can be found by simply re-traversing among the leaves of the tree at the last level. However, such analysis appears feasible only when the per unit cost of non-renewable resource(s) along with their quantities saved, the cash flows involved with completion of activities, and the project's bonus are of comparable magnitudes.

7. Conclusion

This paper describes the non-regular measure (NPV) implementations of the breadth-first and best-first algorithms for multi-mode projects, namely Breadth-first NPV and Best-first NPV. Both the algorithms yield an optimal solution, for which theoretical proofs are included. The experimental results of tests on problem sets generated using PSPLIB problem sets as base problems are presented and discussed. Being extremely difficult problems to solve, no exact solution approach for these problems considering renewable, as well as, non-renewable resources, appears in the literature to the best of our knowledge. Additionally, we consider cash inflows and outflows which may be associated with each time period of the activity in any of its mode that it is performed, and a bonus for up to four time periods of the projects' due date.

Being extremely difficult problem sets to solve, no exact solution approaches for non-regular measures for MM-RCPSP exist in literature. Standard problem sets for non-regular measures are also missing from libraries of such problems. We test the algorithms for non-regular performance measures on payment schedules generated using our own generator, where the PSPLIB problem instances are used as the base problems. The breadth-first algorithm for non-regular measures is also extensible to multi processor SMP implementation. The extension of Breadth-first NPV over shared memory multi processor compute-clusters has not been tested by us and appears to be of immense interest for future research, especially as unlike for regular measures where the run times between Breadth-first and Best-first differ by a large scale, in non-regular measures the difference appears to be less.

Other interesting further research directions include incorporation of improvements in pruning rules, consideration of shared renewable resources, incorporation of resource vacations/withdrawals, dynamic and stochastic models of problem instances, and development of hybrid algorithms incorporating strengths of heuristics, metaheuristics, and exact solution algorithms.

8. References

Dayal, M., and Verma, S. Breadth-first and Best-first Exact Procedures for Regular Measures of the Multi-mode RCPSP. Working Paper No. WP2014-10-04, Indian Institute of Management Ahmedabad, Research and Publication Department, 2014.

Dayanand, N., and Padman, R., 1997. On modelling payments in projects. Journal of the Operational Research Society, 48(9): 906-918.

Dhavale, N. P., Verma, S., and A. Bagchi. Scheduling Partially Ordered Jobs Under Resource Constraints To Optimize Non-Regular Performance Measures. Working Paper No. WP2003-07-03, Indian Institute of Management Ahmedabad, Research and Publication Department, 2003.

Icmeli, O., and Erenguc, S. S., 1996. A branch and bound procedure for the resourceconstrained project scheduling problem with discounted cash flows. Management Science, 42(10): 1395-1408.

Sabzehparvar, M. and S. Seyed-Hosseini, 2008. A mathematical model for the multi- mode resource-constrained project scheduling problem with mode dependent time lags. The Journal of Supercomputing, 44(3): 257-273.

Ulusoy, G., S. S. Funda, and S. Sahin, 2001. Four Payment Models for the Multi-Mode Resource Constrained Project Scheduling Problem with Discounted Cash Flows. Annals of Operations Research, 102(1-4).

Vanhoucke, M., Demeulemeester, E., and W. Herroelen, 2001. On maximizing the net present value of a project under renewable resource constraints. Management Science, 47(8): 1113-1121.

Waligóra, G., 2008. Discrete-continuous project scheduling with discounted cash flows-A tabu search approach. Computers & Operations Research, 35(7): 2141-2153.