Optimally Scheduling Resource Constraint Project Using SAS/OR®

Jeff Cai, Amgen Inc., Thousand Oaks, CA

ABSTRACT

This paper shares with SAS users an approach to effectively distribute programming resources in a clinical study project or among several projects by using procedures and macros in SAS/OR®.

Considering some activities in a project may have precedence and time constraints, and each programmer may also have different time constraints, such as different start date and quit date on a project, personal vacation, etc., one effective method is to use constraint programming procedure CLP to optimally schedule these activities for project management. By incorporating a specific calendar, user can estimate the project target date with the constraint resource, or forecast the required programming resource with the constraint target date.

Plot procedures NETDRAW and GANTT for visualizing activity data are described in this paper as well.

Keywords: SAS/OR, CLP, NETDRAW, GANTT, CSP, Project Management

INTRODUCTION

From SAS 9.1, the constraint programming procedure `PROC CLP` was introduced in SAS/OR® for solving constraint satisfaction problems (CSP) with linear, logical, global, and scheduling constraints. Here CSP can be described by a set of variables *X*, a set of possible values *D* for each variable, and a set of constraints *C* between the variables. Different methods of constraint programming are used to determine whether there exists an assignment of values to variables, so that all the constraints are satisfied.

There are two modes in the context of the CLP procedure to solve two different type of constraint satisfy problems: standard CSPs and scheduling CSPs. The standard mode is to resolve linear constraints, all-different constraints, and array constraints. For example, the Send More Money problem and the Eight Queens problem are two standard CSPs. The scheduling mode can process some scheduling constraints such as temporal constraints and resource constraints.

In scheduling mode, the variables are referred to as activities and the solution is referred to as a schedule. In a statistical programming environment of clinical study project, activities could be but not limited to all sorts of analyses in study level or product level, such as UAT, CSR, PSR, DMC, ISS/ISE, etc. Even in each analysis, we can divide the project into activities such as SDTM creation, ADaM generation, TLG programming, etc. Constraints in a project could be the timeline of all activities or limited resources such as available SAS programmers. Based on the constraint time, or constraint programming resource, or both, a schedule as the solution can be analyzed by `PROC CLP`.

ACTIVITY DATA WITH CONSTRAINTS

In this paper, a real-life clinical statistical programming project is used as an example to show the usage of resource-constraint scheduling with temporal constraints by `CLP` procedure. Table 1 displays the programming activities, precedence constraints among activities, duration of each activity, and resource constraints. Each activity belongs to a group such as SDTM, ADaM, TLG, etc. We can consider it a milestone when completing each group of activities. Each group has been added a null start activity and null end activity to indicate the milestone. The duration for such null activities is zero. The numbers in the "Resource" column represent SAS programmers such as programmer P1, programmer P2, and programmer P3. Most activities can be performed by any programmer ("P1, P2, P3"). Some specific activity can be assigned to specific programmer. For example, in the table, activity "DM" is assigned to programmer P1. The "Status" column is the status of activity completion. If it is completed, it will be shown as "FINISH". We name this programming activity dataset as `PgmAct`.

The activity dataset accepted by CLP procedure requires at least two default variables: one is `_ACTIVITY_` which determines the activity, and the other is `_DURATION_` which determines the activity duration. Without the required variables CLP procedure can't execute. Default variable `_SUCCESSOR_` in the activity dataset defines precedence-type relationships between activities. The following code creates the required activity dataset `ActData` from dataset `PgmAct`. Partial observations from dataset `ActData` are displayed in Table 2.

```
proc sql noprint; ❶
  select activity into :sdtm separated by ','
  from PgmAct where group='SDTM'
  ;
  select activity into :lkup separated by ','
```

```
  from PgmAct where group='LKUP'
  ;
  select activity into :adam separated by ','
  from PgmAct where group='ADAM'
  ;
  select activity into :tlg separated by ','
  from PgmAct where group='TLG'
  ;
quit;
data PgmAct; ❶
  length predecessors $1000;
  set PgmAct;
  if activity='E_SDTM' then predecessors="&sdtm";
  else if activity='E_LKUP' then predecessors="&lkup";
  else if activity='E_ADAM' then predecessors="&adam";
  else if activity='E_TLG' then predecessors="&tlg";
  order=_n_;
run;
proc sort data=PgmAct;
by activity;
run;
data pred(keep=_ACTIVITY_ _SUCCESSOR_);❷
 length _ACTIVITY_ _SUCCESSOR_ $11;
 set PgmAct(rename=(activity=_SUCCESSOR_));
 predecessors=compress(predecessors);
 if predecessors > "" then do;
   nPred=length(predecessors)-length(compress(predecessors,','))+1;
   do i=1 to nPred;
      _ACTIVITY_=scan(predecessors,i);
      output;
   end;
 end;
run;
proc sort data=pred;
by _ACTIVITY_;
run;
data ActData(keep=_ACTIVITY_ _SUCCESSOR_ _DURATION_ _DESC_ order); ❷
  merge PgmAct(rename=(activity=_ACTIVITY_ description=_DESC_))
        pred;
  by _ACTIVITY_;
  _DURATION_=duration;
  label _ACTIVITY_   = 'Activity'
        _SUCCESSOR_  = 'Successor Activity'
        _DESC_       = 'Description of Activity'
        _DURATION_   = 'Duration of Activity'
        ;
run;
proc sort data=ActData;
by order;
run;
```

The PROC SQL statements and data step marked by ❶ replace “All activities from xxx” with all real activity names from group xxx. Here xxx may be “SDTM”, “LKUP”, “ADAM”, or “TLG”. Data steps marked by ❷ transform records from `predecessors` to `_SUCCESSOR_`.

To visualize the activity data, a network diagram of the activities is drawn by PROC NETDRAW in SAS/OR®. In Figure 1, nodes represent the activities, and arcs represent the precedence relationships among the activities.

```
title h=3  j=l ' Network Diagram: Statistical Programming Activities';
proc netdraw data=ActData graphics ; ❸
     actnet / act=_ACTIVITY_ succ=(_SUCCESSOR_)
              height=1.5 compress arrowhead=2
              carcs=blue; ❹
run;
```

Option `data=` marked by ❸ is for the activity dataset used by `PROC NETDRAW` to produce a network diagram. Option `act=` in `ACTNET` statement marked by ❹ specifies the variable in the activity dataset that names the nodes. Here we assign variable `_ACTIVITY_` as the nodes to option `act=`. Option `succ=` specifies all the immediate successors of the node specified by `_ACTIVITY_`. Here we assign variable `_SUCCESSOR_` to option `succ=`.

CLP PROCEDURE

When the activity dataset with required variables is ready, we can call PROC CLP to generate an optimal scheduling plan.

```
proc clp actdata=actdata scheddata=scheddata;❺
   schedule edgefinder=first finish=60;❻
    resource (P1-P3);❼
    requires &req;❽
run;
%put &_ORCLP_;❾
```

Option `actdata=` in `proc clp` statement❺ accepts input dataset that defines the activities and temporal constraints. If there is no activity dataset assigned, the activities and temporal constraint should be specified in the `activity` statement, which is one of the statements in CLP procedure not shown in the above code. Option `scheddata=` identifies the output dataset that contains the scheduling solution to the resource constraint project.

Option `edgefinder=` in the `schedule` statement❻ invokes the edge-finding consistency routine for scheduling problem. Edge-finding consistency routine can determine each activity's priority based on the same resource or set of resources so as to form the processing order of a set of activities. Specifying `edgefinder=first` here is to detect whether a given activity must be processed first in a set of activities that require the same resource or set of resource. Option `finish=` is to specify the finish time for the schedule. Here `finish=60` means whether a scheduling solution exists when the project is required to be finished within 60 days.

The `resource` statement❼ specifies all resources allocated to the activities. In our example, available resources are three programmers (P1-P3).

The `requires` statement specifies the available resources associated with each activity. Macro variable ***req***❽ is the specification generated from dataset `PgmAct` by the following code:

```
** Create macro varible for REQUIRES statement in CLP procedure **;
proc sort data=PgmAct out=requires;
by resource activity;
where resource > '';
run;
data _null_;
  length requires $2000;
  set requires end=last;
  by resource activity;
  retain requires '';
  if first.resource then requires=trim(requires)||' (';
  requires=trim(requires)||' '||strip(activity);
  if last.resource then
     requires=trim(requires)||') = ('||strip(resource)||')';
  if last then call symput('req',strip(requires));
run;
```

Macro `_ORCLP_` is automatically generated by the `CLP` procedure to indicate the completion status of the procedure. In the log file, we can find the message "`STATUS=SUCCESSFUL SOLUTIONS_FOUND=1`" printed by the macro `_ORCLP_`, which indicates the `CLP` procedure executes successfully and there is one scheduling solution to finish this project in 60 days by three programmers.

SCHEDULING SOLUTION GENERATED FROM CLP PROCEDURE

The "product" generated by `CLP` procedure is the dataset `scheddata`❺. In dataset `scheddata`, besides the variable `ACTIVITY` and `DUR`, there are new variables: `SOLUTION`, `START`, `FINISH`, `P1`, `P2`, and `P3`. Since there is only one solution

in the example, values of SOLUTION are all 1. START and FINISH are scheduled start day and end day. P1, P2, and P3 are flags of the activities assigned to the programmers. Here we combine P1, P2, P3 into one variable RESOURCE and display the data in Table 3. Although this project has 60 day time-constraint❻, the optimal scheduling solution generated by the CLP procedure only requires 49 days.

```
data scheduling(keep=activity dur start finish resource);
  set scheddata;
  length RESOURCE $2;
  if P1=1 then resource='P1';
  else if P2=1 then resource='P2';
  else if P3=1 then resource='P3';
run;
proc sort data=scheduling;
 by start finish resource;
run;
```

To visualize the scheduled programming activity in dataset scheduling, GANTT procedure is used to generate a Gannt chart segmented by the scheduled programming activity of each programmer (Figure 2).

```
** Create variable _pattern for PATTERN= option in GANTT procedure***;
** _pattern variable is used to identify the pattern for drawing the bar**;
data sched;
 set scheduling;
 _pattern=_n_;
run;
proc sort data=sched;
     by resource start;
run;
*** Add segmt_no variable to identify the number of segments ***;
data newsched;
 set sched;
 retain segmt_no;
 if resource ne lag(resource) then segmt_no=1;
 else segmt_no = segmt_no + 1;
 output;
run;
***create reference line data for REF= option in GANTT procedure ***;
proc sql noprint;
    select distinct start into : ref separated by ' '
    from scheduling
   where activity in ('E_SDTM' 'E_ADAM' 'E_TLG' 'E_LKUP');
quit;
*** Create label data ***;
data labels;
    _y=-1;
    _lvar="activity";
    _xvar="start";
    _flabel="";
    _hlabel=1;
    _yoffset = -.2;
run;
pattern1 v=s r=25;
title h=3  j=l ' Gantt Chart: Scheduled Programming Activities by Each Programmer';
proc gantt data=newsched labdata=labels graphics;
  id resource;
  chart / ss=start sf=finish compress labsplit='.' scale=2
          nolegend nojobnum skip=3 ref=&ref; ❿
run;
```

In the Gantt chart, every row corresponds to a programmer. Every bar on each row consists of multiple segments, and every segment represents an activity. The x-axis lists the schedule timeline in days, and therefore the start and end of each segment represent the start and finish times for each activity. We notice there are four light blue reference lines in Figure 2, which are created by ref= option in the chart statement❿. We use these reference lines to indicate different project milestones:

completion of SDTM, lookup tables, ADaM, and TLG. The chart is very efficient to track the progress of multi jobs multi programmers at the same time.

DISCUSSIONS

Scenario 1

If we know the exact programming activities and number of programmers available for the project, how can we know the optimal timeline to arrange each programmer on the project? That is, how to finish the project with minimum time under the available resource? An approach already has been shown to optimize the resource-constraint scheduling in this paper. In the example, three programmers are schedule to finish the project in 49 days.

Scenario 2

If the project timeline is very tight and it is required to be finished in less time, what we should do? Go back to our example in this paper, and we can ask if we can finish the project in 30 days by three programmers. To test this case, we replace the option `finish=60`❻ with `finish=30`, and execute the CLP procedure again. The string of macro variable `_ORCLP_`❾ will show there is no solution to schedule the project based on the current resource constraint and time constraint. In order to resolve this problem, we can increase the number of programmer from three to four to achieve a solution. By this way we can forecast the resource need for oncoming project.

Scenario 3

In the middle of project progress, if any programming activity is added or dropped, or if any programmer joins in or quits, or if the target date is changed, how can we reschedule the timeline and reallocate the resource on the project? In the dataset `PgmAct` listed in Table 1, we have variable `status` which identified the status of each activity. If an activity is finished, "FINISH" will be updated in the variable `status` and the duration of the activity will be set to zero. We can update the programming resource and project target day by updating `schedule`❻, `resource`❼, and `requires`❽ statements in the `CLP` procedure. A scheduling solution can be achieved any time by rerunning the `CLP` procedure. By incorporating the actual schedule for finished activities in the Gantt chart, the project overall status can be easily monitored.

Scenario 4

Finally, a SAS macro `%SASTOMSP` in SAS/OR® can be used to convert the scheduled SAS dataset into a form that is readable by Microsoft Project. Each programmer can get their individual task schedule in the format of Microsoft Project.

CONCLUSION

`CLP` procedure is an effective approach to optimally scheduling resource and time constraint programming projects. By using this approach, ongoing project resource can be rescheduled and reallocated, and resource requirement for the future projects can be forecasted. The visualized schedule data in Gantt charts can display complex timelines and keep project status updated.

REFERENCES

SAS Institute Inc. *SAS/OR 9.2 User's Guide: Constraint Programming*. Cary, NC
Tsang, Edward (1993). *Foundations of Constraint Satisfaction.* Academic Press.

ACKNOWLEDGEMENTS

I would like to thank Dan DiPrimeo and Alan Nakamoto for reviewing this paper.

CONTACT INFORMATION

Your comments and questions are valuable and appreciated. Author can be reached at

Jeff Cai
Amgen Inc.
1120 Veterans Blvd
South San Francisco, CA 94080 U.S.A.
Email: jeff.cai@amgen.com

Table 1 Programming Activities in A Typical Statistical Analysis (`PgmAct`).

Activity	Group	Description	Predecessors	Duration	Resource	Status
S_PROJECT		Start of Project		0		FINISH
S_SDTM		Start of SDTM Creation	S_PROJECT	0		FINISH
DM	SDTM	Demographics	S_SDTM	2	P1	
AB	SDTM	Antibody	DM	1	P1, P2, P3	
AE	SDTM	Adverse Events	DM	2	P1, P2, P3	
CM	SDTM	Concomitant Medications	DM	1	P1, P2, P3	
CO	SDTM	Comments	DM	1	P1, P2, P3	
DF	SDTM	Disorder Findings	DM	3	P1, P2, P3	
DS	SDTM	Disposition	DM	3	P1, P2, P3	
EG	SDTM	ECG	DM	1	P1, P2, P3	
EX	SDTM	Exposure	DM	2	P1, P2, P3	
IE	SDTM	Inclusion/Exclusion Criteria	DM	1	P1, P2, P3	
LB	SDTM	Laboratory Test Results	DM	1	P1, P2, P3	
MH	SDTM	Medical History	DM	1	P1, P2, P3	
PE	SDTM	Physical Exams	DM	1	P1, P2, P3	
QS	SDTM	Questionnaires	DM	1	P1, P2, P3	
SX	SDTM	Surgical and Procedural Interventions	DM	2	P1, P2, P3	
TR	SDTM	Tumor Results	DM	3	P1, P2, P3	
TU	SDTM	Tumor Identifications	DM	3	P1, P2, P3	
VS	SDTM	Vital Signs	DM	1	P1, P2, P3	
E_SDTM		End of SDTM Creation	All Activities from SDTM	0		
S_LKUP		Start of Lookup Tables Creation	S_PROJECT	0		
LKUP_AE	LKUP	AE Lookup Table	S_LKUP	1	P2	
LKUP_CM	LKUP	CM Lookup Table	S_LKUP	1	P2	
LKUP_MH	LKUP	MH Lookup Table	S_LKUP	1	P2	
E_LKUP		End of Lookup Tables Creation	All Activities from LKUP	0		
S_ADAM		Start of ADaM Creation	S_PROJECT	0		
ADSL	ADAM	Subject Level Information Analysis Data	S_ADAM, AE, CM, DF, DM, DS, EX, LB, PE, QS, SX, TR, TU, VS	4	P3	
AAB	ADAM	Antibody Analysis Data	ADSL, AB	1	P1, P2, P3	
AAE	ADAM	Adverse Events Analysis Data	ADSL, AE, LKUP_AE	2	P1, P2, P3	
ADS	ADAM	Disposition Analysis Data	ADSL, DS	3	P1, P2, P3	
ACM	ADAM	Concomitant Medications Analysis Data	ADSL, CM, LKUP_CM, DF	1	P1, P2, P3	
AEG	ADAM	ECG Analysis Data	ADSL, EG	1	P1, P2, P3	
AEX	ADAM	Investigational Product Administration Analysis Data	ADSL, EX	3	P1, P2, P3	
AVS	ADAM	Vital Signs/Physical Examination Analysis Data	ADSL, VS, PE	1	P1, P2, P3	
ALB	ADAM	Laboratory Analysis Data	ADSL, LB, EX	3	P1, P2, P3	
AQS	ADAM	Questionnaire Analysis Data	ADSL, QS	1	P1, P2, P3	
AMH	ADAM	Medical and Surgical History	ADSL, MH, LKUP_MH	1	P1, P2, P3	
ALS	ADAM	Target and Non-target Lesion Analysis Data	ADSL, TR, TU	2	P1, P2, P3	
AEFF	ADAM	Efficacy Summary Analysis Data	ADSL, ALS, DF, SX, TR, TU	3	P1	
ASAF	ADAM	Safe Summary Analysis Data	ADSL, AAE, ACM, AMH, AEX, AVS, ALB, SX	3	P2	
E_ADAM		End of ADaM Creation	All Activities from ADAM	0		

Continued

Activity	Group	Description	Predecessors	Duration	Resource	Status
S_TLG		Start of TLG Creation	S_PROJECT	0		
g_disp	TLG	Subject Disposition Graphs	ADS	1	P1, P2, P3	
g_inf_time	TLG	Infusion Times Graphs	AEX	1	P1, P2, P3	
g_dose	TLG	Cumulative Dose Bar Charts	ASAF	3	P1, P2, P3	
g_survival	TLG	Kaplan-Meier Plots	AEFF	3	P1, P2, P3	
g_infrea	TLG	Subject Incidence by Infusion Number	AAE, AEX	2	P1, P2, P3	
g_lab	TLG	Median Laboratory Value	ALB	2	P1, P2, P3	
g_vital	TLG	Median Vital Signs	AVS	2	P1, P2, P3	
l_sae	TLG	Serious Adverse Events	AAE, ADS	1	P1, P2, P3	
l_conmed	TLG	Subject Listing of CM Considered as Invest. Drug Infusion Reaction	ACM, AAE, AEX	1	P1, P2, P3	
t_death	TLG	Death Summary	ADSL, AAE, ADS	1	P1, P2, P3	
t_charact	TLG	Baseline Disease Characteristics	ADSL, AEG, ALB, ALS	2	P1, P2, P3	
t_sum_pana	TLG	Summary of the Primary Analysis of Survival Time	ADSL, AEFF	3	P1, P2, P3	
t_pfsurtime	TLG	Sensitivity Analysis for the Log-rank Test of Survival	ADSL, AEFF	3	P1, P2, P3	
t_anatreat	TLG	Univariate Cox Proportional Hazards Analysis	ADSL, AEFF	3	P1, P2, P3	
t_aes	TLG	Summary of Adverse Events	ADSL, AAE	2	P1, P2, P3	
t_infrea	TLG	Incidence of Infusion Reactions	ADSL, AAE, AEX	3	P1, P2, P3	
t_skintox	TLG	KM Analysis of Integument Toxicity	ADSL, ASAF	2	P1, P2, P3	
t_labshift	TLG	Lab Grade Shifts	ADSL, ALB	2	P1, P2, P3	
t_vitals	TLG	Summary of Vital Signs	ADSL, AVS	1	P1, P2, P3	
t_ab	TLG	Summary of Antibody Incidences	ADSL, AAB	2	P1, P2, P3	
t_ab_paor	TLG	Primary Analysis of Objective Response	ADSL, AAB, AEFF	2	P1, P2, P3	
E_TLG		End of TLG Creation	All Activities from TLG	0		
E_PROJECT		End of Project	E_LKUP, E_SDTM, E_ADAM, E_TLG	0		

Table 2 Partial Observations in the Activity Dataset (`ActData`).

ACTIVITY	_DESC_	_SUCCESSOR_	_DURATION_
S_PROJECT	Start of Project	S_ADAM	0
S_PROJECT	Start of Project	S_LKUP	0
S_PROJECT	Start of Project	S_SDTM	0
S_PROJECT	Start of Project	S_TLG	0
S_SDTM	Start of SDTM Creation	DM	0
DM	Demographics	AB	2
DM	Demographics	ADSL	2
DM	Demographics	AE	2
DM	Demographics	CM	2
DM	Demographics	CO	2

Figure 1 Network Diagram for Programming Activities.

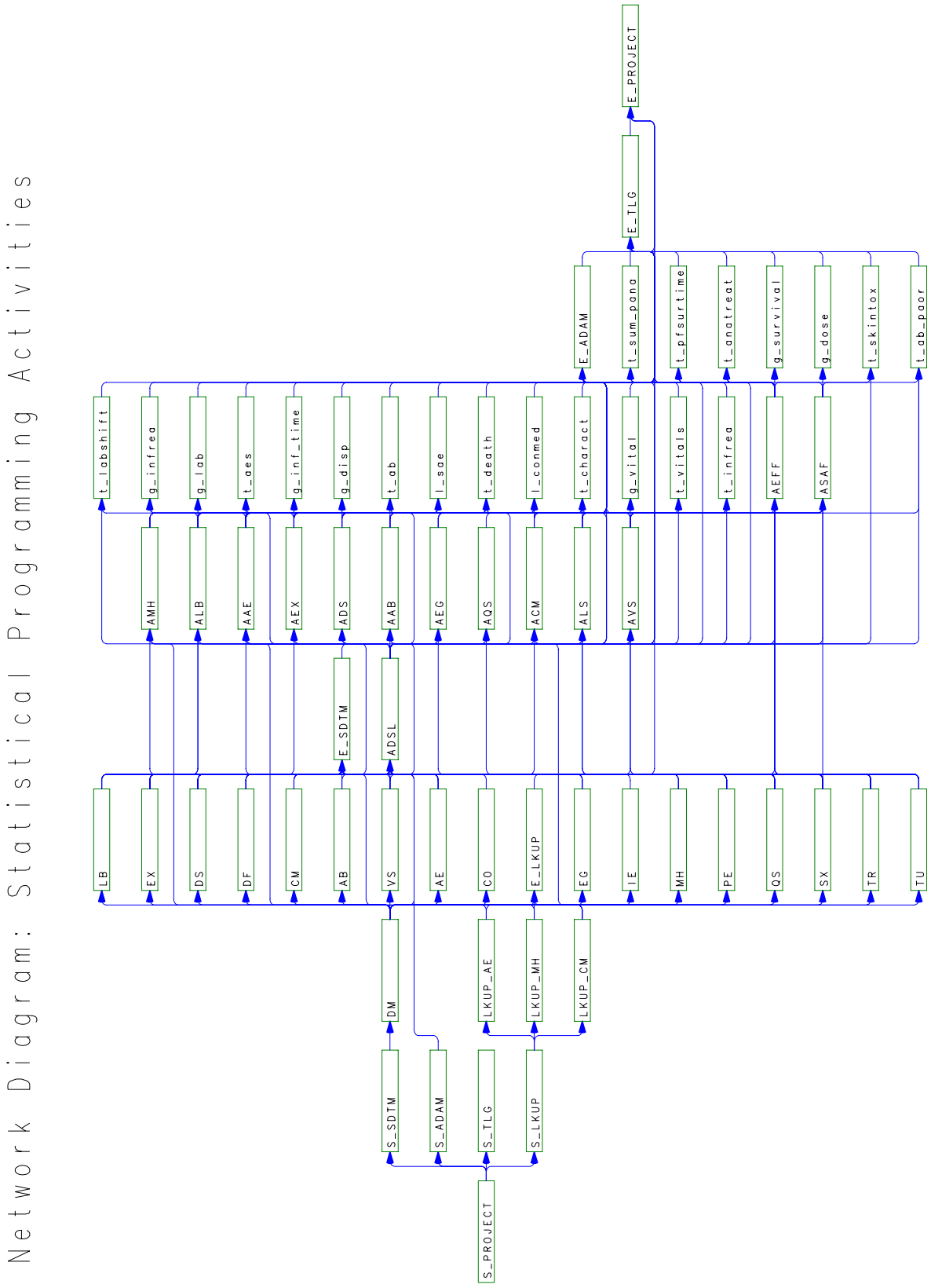

Table 3 A Scheduling Solution Generated by CLP Procedure (`scheduling`).

ACTIVITY	DUR	START	FINISH	RESOURCE
S_PROJECT	0	0	0	P1
S_SDTM	0	0	0	P2
S_ADAM	0	0	0	P3
S_LKUP	0	0	0	P3
S_TLG	0	0	0	P3
LKUP_CM	1	0	1	P1
LKUP_AE	1	0	1	P2
LKUP_MH	1	0	1	P3
E_LKUP	0	1	1	P3
DM	2	1	3	P1
IE	1	3	4	P2
AE	2	3	5	P1
DF	3	3	6	P3
MH	1	4	5	P2
VS	1	5	6	P2
TU	3	5	8	P1
SX	2	6	8	P2
EX	2	6	8	P3
CM	1	8	9	P1
CO	1	8	9	P3
TR	3	8	11	P2
QS	1	9	10	P1
AB	1	9	10	P3
EG	1	10	11	P1
LB	1	10	11	P3
PE	1	11	12	P1
DS	3	11	14	P3
E_SDTM	0	14	14	P3
ADSL	7	14	21	P1
AMH	1	21	22	P1
AVS	1	21	22	P2
AAE	3	21	24	P3
AQS	1	22	23	P2
g_vital	2	22	24	P1
AAB	1	23	24	P2
AEG	1	24	25	P2
AEX	3	24	27	P1
ADS	4	24	28	P3
ALS	2	25	27	P2
ACM	1	27	28	P2
AEFF	5	27	32	P1
t_ab	2	28	30	P2
ALB	4	28	32	P3
g_infrea	2	30	32	P2
g_disp	1	32	33	P3
t_aes	2	32	34	P2
g_survival	4	32	36	P1
t_sum_pana	3	33	36	P3
t_infrea	3	34	37	P2
t_anatreat	3	36	39	P3
t_labshift	4	36	40	P1
t_pfsurtime	3	37	40	P2
ASAF	7	39	46	P3
l_sae	1	40	41	P2
g_lab	2	40	42	P1
t_death	1	41	42	P2
t_charact	2	42	44	P2
t_ab_paor	4	42	46	P1
t_vitals	1	44	45	P2
l_conmed	1	45	46	P2
E_ADAM	0	46	46	P2
g_inf_time	1	46	47	P2
t_skintox	2	46	48	P3
g_dose	3	46	49	P1
E_TLG	0	49	49	P3
E_PROJECT	0	49	49	P3

Figure 2 Gantt Chart for Scheduled Programming Activities.