Who Should Own Float? Mitigating Delays by Float Pre-Allocation

**Gunnar Lucko, Ph.D., Associate Professor of Civil Engineering,
and Yi Su, Graduate Research Assistant, Catholic University of America**

Abstract

Float reduces risk by protecting against delays in network schedules. But who owns it and how much remains a contentious issue. This paper presents results of a National Science Foundation-funded research on allocating project float – between the sum of raw durations and the contract deadline – to the critical path. A mathematical model from political decision-making has inspired how to fairly allocate it so that the critical participants can reach an evenly low risk level. Its contribution to the body of knowledge is threefold: First, various political apportionment methods are reviewed and their characteristics are summarized. Second, a methodology for fairly distributing float is created, which includes ways to calculate the quota (which is used in apportionment methods) based on an exponent of size. Third, simulations of a randomized schedule are performed to validate the new approach. Results indicate that using the square-root based allocation achieves the desired fairness over apportionment with equal or proportional shares.

1. Introduction

Construction projects often incur delays, which are "an act or event that extends the time required to perform tasks under a contract" (Stumpf 2000, p. 32). The most crucial elements that contribute to delays are those activities on the critical path. To protect these activities and guarantee that the project will finish as planned, a general approach places an aggregated buffer at the end of the critical path, called *project buffer* by critical chain project management (Tukel *et al.* 2006). But this approach radically cuts activity durations, assuming that they harbor individual float, returns half of it as a single large end buffer, and then accepts that said 'block' is consumed fully without tracking by whom. None of these features properly protects a project (Herroelen and Leus 2001) and some are downright counter-productive. To overcome this paradigm, this research revises the central question of *who owns the float* – a controversial issue in research and practice – to a more appropriate *how can available project float be used most beneficially by allocating it to activities that by definition have none and determining how much each receives*?

1.1 Similarities between Political Apportionment and Float-Preallocation

Motivated by the fact that among "bedrock principles in a democracy is the equal consideration of the preferences and interests of all citizens" (Verba 2003, p. 663), political apportionment "is the problem of determining how to divide a given integer number of representatives or delegates proportionally among given constituencies according to their respective sizes" (Balinski and Young 1977, p. 607). Interestingly, such apportionment shares characteristics with the float pre-allocation problem per Table 1: First, both methods are motivated by the goal to fairly divide a limited good (seats or float). Second, the object to distribute, seats or float, has non-negative integer values. Third, both approaches must (round a number with fractional part to the nearest integer value) to guarantee that the final apportionment is the correct integer sum. Neither fractional seats nor partial days (the major time unit of construction project scheduling) of float exist in reality. Inspired by the successful political apportionment approaches, this paper therefore will develop an analogous scheduling approach to distribute the project float to every critical activity. Project float is defined as the buffer from calculated (based on raw durations) to contractually required project finish, i.e. it resides after the raw project finish. Three research objectives are set to realize this goal:

Table 1: Analogies between Political Apportionment and Float Preallocation

Similarities	Float Preallocation	Political Apportionment
Why apportionment (Goal)	Fairness for distributing float	"[E]qual influence over government policy across citizens" (Verba 2003, p. 663)
What is equalizing (subject)	Float/time/ability of excusable delay when making schedule	Seats/power of voting when making policy
How to apportion (method)	Rounding, because scheduling results typically rounds to integer days, while software allows integer minutes	Rounding, because no fractional seats are allowed

1. Reviewing different rounding methods that are used in political voting and extracting their pertinent features;
2. Creating a methodology to integrate apportionment methods and their rounding into pre-allocating project float;
3. Conducting a simulation of a sufficiently complex schedule to validate these methods and compare their results.

2. Political Apportionment Methods

In the literature, political apportionment is performed with six major methods: Greatest divisor; smallest divisor; geometric mean; harmonic mean; arithmetic mean; and largest remainder, respectively. From a general view, these methods can be further classified into largest remainder and divisor methods, where the latter require selecting the divisor "based on the generalized power mean for specific parameter values" (Jones and Wilson 2010, p. 343). For illustration, consider a small example of apportioning votes of four parties to a house that contains 45 seats. Their votes, which represent their relative 'size' or importance, are 61 for A, 48 for B, 20 for C, and 7 for D, respectively.

2.1 Largest Remainder Method

Also known as the Hamilton method, its process is as follows: First calculate the proportion of each party of the total votes (e.g. A: 61 / (61 + 48 + 20 + 7) = 44.85%). Then multiply this ratio with the total seat to obtain the unrounded seats, which is known as its quota (e.g. the quota of A is 20.18). Next round down the quota to an integer (e.g. 20 for A). At this stage, the tentative seats per party are 20 for A, 15 for B, 6 for C, and 2 for D, so that 43 seats have been distributed. The leftover two seats are given to the parties with the largest remainders per Table 2, B (0.88) and C (0.62). The final result of this apportionment is 20 for A, 16 for B, 7 for C, and 2 for D. But a fallacy called the 'Alabama Paradox' exists for this method, that increasing the number of seats in the house "has caused the smallest party to lose a seat" (Wiseman 2015) in certain cases, which is a rounding-related counterproductive result. In this example, it will occur if a 63 seat house adds one; {28, 22, 10, 3} becomes {29, 23, 9, 3}, i.e. C will lose one seat.

Table 2: Results of Distributing Seats for Four Parties with Different Apportionment Methods

	Target Proportion						Quota Unrounded				Seats			
Method Name	A	B	C	D	**Total Seats**	**Divisor**	A	B	C	D	A	B	C	D
Largest Remainder	61	48	20	7	45	N/A	20.18	15.88	6.62	2.32	20	16	7	2
Greatest Divisor						2.90	21.03	16.55	6.90	2.41	21	16	6	2
Smallest Divisor						3.20	19.06	15.00	6.25	2.19	20	15	7	3
Geometric Mean						3.02	20.49	15.49	6.48	2.45	20	16	7	2
Harmonic Mean						3.02	20.49	15.48	6.46	2.40	20	16	7	2
Arithmetic Mean						3.07	19.87	15.64	6.51	2.28	20	16	7	2

2.2 Divisor Methods

Divisor methods require finding a proper divisor and apply different rounding rules, depending on the particular method. A greatest divisor is used in the D'Hondt / Jefferson method. Its process is to "choose a common divisor λ, and for each state compute p_i / λ and round *down* to the nearest integer" (Balinski and Young 1978, p. 279, emphasis in original). The smallest divisor in the Adams method is similar to the greatest divisor method, but rounds up. The geometric mean in the Huntington-Hill method – used in the U.S. House of Representatives – initially rounds down the quota q of each party, calculates the geometric mean as the square root of q times q + 1, and judges as follows: If q is smaller than the geometric mean, then apportioned seats are the rounded up geometric mean, else the rounded down geometric mean. The harmonic mean in the Dean method is similar, but employs the harmonic mean. The Sainte-Laguë / Webster method similarly uses the arithmetic mean. If necessary, the divisor λ must be iteratively adjusted for all initial quotas until the correct total number of seats is reached. The summaries of these political apportionment methods and their differences in terms of numerical results in Tables 2 fulfill Research Objective 1.

3. Methodology

3.1 Float Preallocation Model

The overall structure of the float apportionment – here it is a pre-allocation, because it is applied before the project starts – as inspired by the aforementioned political voting methods is explained by Table 3. Suppose that a project has n critical activities and their fixed durations are D_1 through D_n. Using these values as the base, three different exponents, 0.0, 0.5, and 1.0, for the relative size (i.e. duration) are tested in this research. Note that for exponent 0.0, all activities will receive an equal proportion of the project float, because any number to the power of zero equals one. The other extreme, exponent 1.0, means that the apportionment is directly proportional to the relative durations of activities. The intermediate case with the square root of duration is more moderate and hypothesized to strike the aforementioned desired fair balance. The different apportionment methods of Table 2 are applied to this example for a variable amount of project float. Output is the set (a_1, …, a_n) as the float pre-allocation, where $\sum_i a_i = FLOAT$.

Table 3: Structure of Float Arrangement Model based on Apportionment Methods

Exponent of Duration	**Target Proportions (Quotas)** Act 1	Act 2	Act 3	…	Act n	**Float to Apportion**	**Apportionment Method**	**Float Apportionment**
0.0	1	1	1	…	1	FLOAT	LR, GD, SD, GM, HM, AM	(a_1, …, a_n)
0.5	$D_1^{0.5}$	$D_2^{0.5}$	$D_2^{0.5}$	…	$D_n^{0.5}$			
1.0	D_1	D_2	D_2	…	D_n			

Note: Different rounding approaches are already included in the apportionment methods.

3.2 Simulation Process

After calculating the numerical value of the float pre-allocation for each critical activity, the next step is to perform a simulation per Table 4, which randomly samples activity durations from a predefined probability distribution. It is then analyzed if the rounded-up integer delay $\lceil d_i - D_i \rceil$ is larger than the arranged float for it (a_i), then one overrun event is counted, and the actual length of that overrun period is calculated. The former is a discrete count, the latter a continuous measurement. After the simulation finished, two plots were generated, which contain a curve for each critical activity, one diagram for the counts of overrun events that exceeded pre-allocated float for said activity, and another diagram for the extent of said overruns of the same activity. This way the actual performance of exponents for the apportionment method (selected from Table 2) can be compared quantitatively. The goal of the simulation that a minimum number of activities incur overruns; and that the overruns themselves remain small. In such case, it is concluded that all of the individual activities have become ‘saturated’ with float so that they rarely if ever become problematic, and that the overall float pre-allocation has been performed in a fair and proactive manner. Fairness here means that all activities are protected against risk so that they reach an even and low level of delays. Having created a methodology that integrates apportionment methods and float pre-allocation fulfills Research Objective 2.

Table 4: Structure of Simulation Process

Run	**Simulation** Act 1	Act 2	Act 3	…	Act n	**Float Arrangement**	**Judgement (Discrete)**	**Judgement (Continuous)**
1	d_{11}	d_{21}	d_{31}	…	d_{n1}	(a_1, …, a_n)	if $\lceil d_i - D_i \rceil > a_i$, then count one overrun event	if $\lceil d_i - D_i \rceil > a_i$, then calculate overrun period
2	d_{12}	d_{22}	d_{32}	…	d_{n2}			
…	…	…	…	…	…			
N	d_{1N}	d_{2N}	d_{3N}	…	d_{nN}			

4. Case Study and Simulation Analysis

A schedule of significant complexity with sixty activities from the Project Scheduling Problem Library (PSPLIB, Kolisch and Sprecher 1996) will be reanalyzed. PSPLIB is a database that contains a multitude of schedule files of different size and complexity to test resource-constrained scheduling approaches and objectively compare optimal algorithms. The selected input file is J60_1, which is the first listed schedule of 60 activities, plus a dummy source and sink node. Appendix 1 lists input data, where bold indicates critical activities. Figure 1 shows its network. The total project (mode) duration based on fixed durations is 77 days. Note that the original input only provides activity name, fixed (mode) duration, and sequential relations. To this the authors have added duration distributions (here a triangular with lower and upper limit at 90% and 150% of the mode duration). The simulation program that has been coded in the MATLAB language can plot the network and support other types of probability distributions, e.g. beta. Two steps are needed: Optimistic and pessimistic project durations (50.4 and 84 days) are calculated from lower and upper limits of activity durations. Their difference is the pre-allocated float, which rounds down to 33 integer days.

4.1 Measuring Performance as Discrete and Continuous Overruns

Following the processes per Tables 3 and 4, the simulation runs systematically increase the amount of project float and record its pre-allocation and performance in discrete and continuous values. It is expected that the more project float is provided, the fewer activities should have absolute or relative overruns. Therefore the curve profiles for both discrete and continuous overruns should decrease. This study uses a standardized way to measure the rate at which this decrease occurs for the different cases: Recording the two project float value at which the profile starts and finishes its decrease (i.e. edges of the 'cliff', taking the mean of these values as the average float, and calculating the minimum, maximum, mean, and standard deviation of the average float for all critical activities and plotting them.

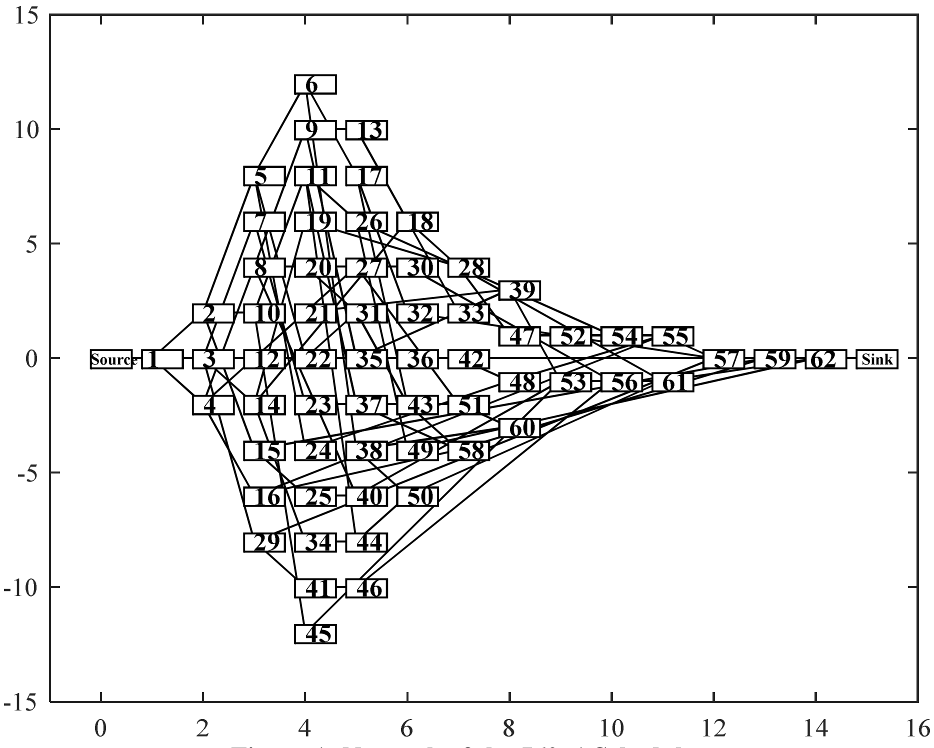

Figure 1: Network of the J60_1 Schedule

4.2 Simulation Output with Largest Remainders Method

A total of 100 simulation runs are performed. The upper and lower three plots in Figure 2 are the discrete overrun counts and continuous overrun periods for the exponents 0.0, 0.5, and 1.0, respectively. Comparing these scenarios shows that only for the square root of duration approach (exponent 0.5) all critical activities stop overrunning at 70 days of project float. Several activities continue overrunning for exponents 0.0 and 1.0, where the float to reach zero overruns for exponent 0.0 is 510 days (outside the chart), but some very short activities never stop overrunning for exponent 1.0; the simulation was stopped at an extreme project float of 1000 days to curb the computational effort. The mean of the average float of all activities is the minimum (11 days) for exponent 0.5, compared to the two other exponents (34 and 44 days, respectively). Note that the reason why all activities start below 100 on the vertical axis is that among 100 simulation runs it will not always the case that a randomized duration exceeds its mode; some durations will be shorter so that no overrun occurs whatsoever. Where a curve reaches zero indicates that from this project float onward an activity has received as much (or more) pre-allocated float as is needed for even the worst possible delay within the assumed probability distribution for its duration. Some activities never receive enough.

4.3 Simulation Output with Webster Method

The time-consuming process of searching iteratively for a suitable divisor is added to the simulation run. A suitable divisor is one that by which each activity quota has been divided so that the apportioned integer values sum to the correct project float. All divisor methods follow the same pattern as the Webster method. They only differ in the way of calculating the divisor and rounding. For brevity, not all simulation results of all divisor methods will be listed. A more comprehensive comparison of their performance characteristics will be provided by future research.

Analogous to the other simulation, 100 runs are performed. Figure 3 shows both discrete and continuous plots. Following the previous pattern, exponent 0.5 is again the optimum case that allows all activities to stop overrunning with the least project float. Compared to the largest remainders method, the Webster method performs better for exponent 0.5, whose maximum of 44 days required project float is substantially lower than a maximum of 50.5 days.

Note that in these simulations, providing 34 days of project float is insufficient to prevent all critical activities from overrunning. This only becomes fulfilled for exponent 0.5 (exponent 0.0 and 1.0 always incur overruns) and then only above 69 days for largest remainder (who still may incur the Alabama Paradox) and 77 days for Webster. This is because some apportionments may incur the so-called Matthew (25:29) effect "[f]or unto every one that hath shall be given…[,] but from him that hath not shall be taken away", which is well known in economics and politics. Such apportionments require more than the optimum quantity so that all of their critical activities stop overrunning.

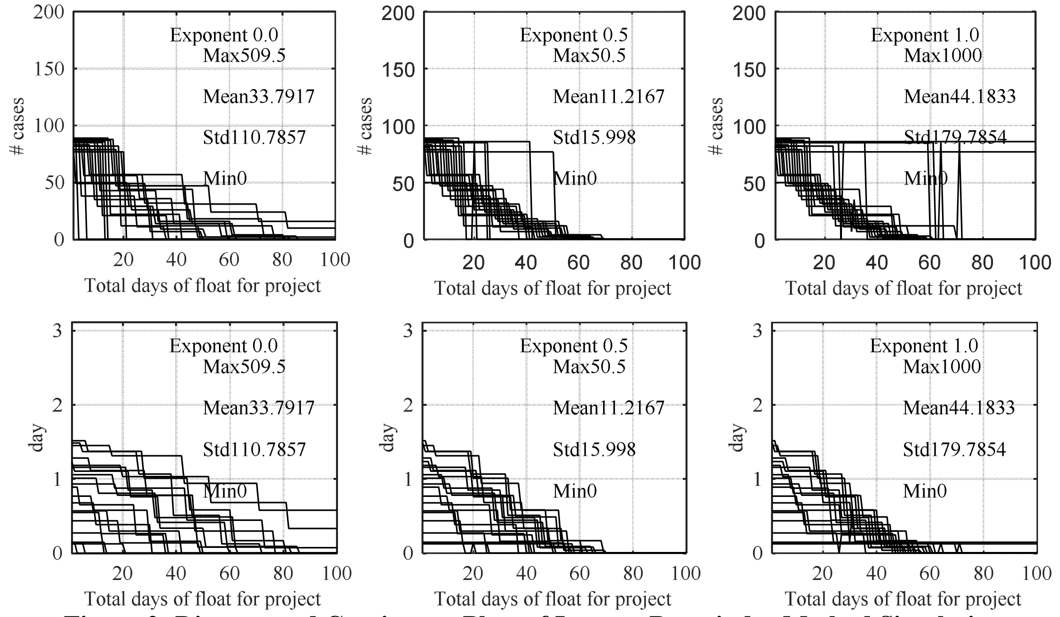

Figure 2: Discrete and Continuous Plots of Largest Remainder Method Simulation

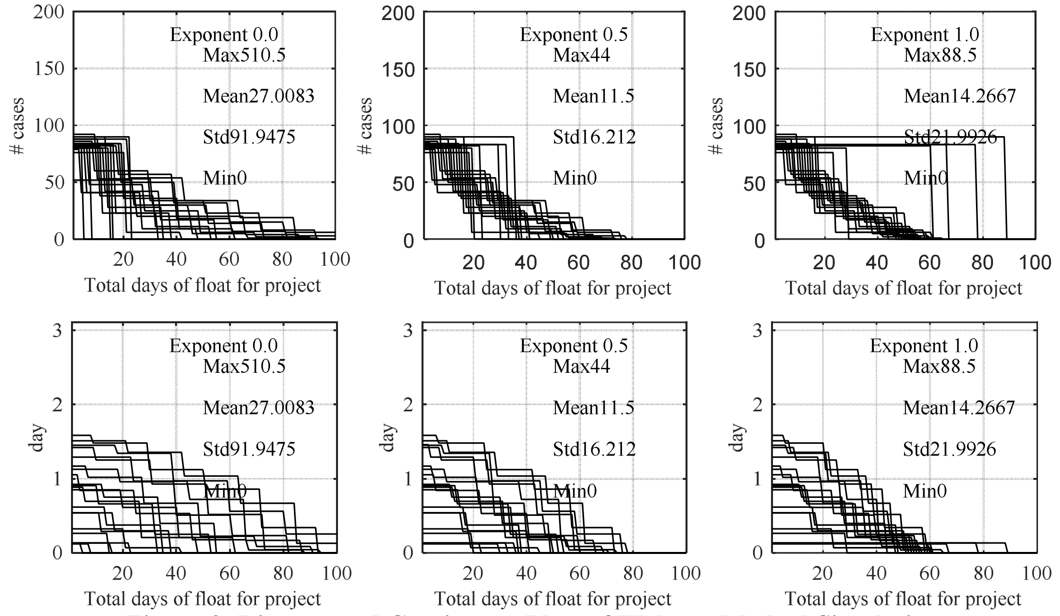

Figure 3: Discrete and Continuous Plots of Webster Method Simulation

Two advantages appear for exponent 0.5: First, the mean of average float overrun decrease is at a minimum, i.e. it represents the most economic way to apportion float. Second, it is fairest to apportion float for all activities, long and short ones – the quota that exponent 0.5 generates is moderate compared to the other exponents. For exponent 0.0, every activity receives exactly the same quota (because any number with exponent zero is one). By using this exponent, all critical activities – regardless of duration – will receive the same pre-allocated float, which clearly is an inefficient apportionment. This has the effect that short activities will stop overrunning sooner, but long activities will require much more float (which they never receive) to stop overrunning. On the other extreme, for exponent 1.0, the quotas of short and long activities will be significantly different. And while it allows long activities to stop overrunning relatively soon, some short activities never receive sufficient pre-allocated float to stop overrunning. Having conducted different simulations and compared their results and performances fulfills Research Objective 3.

5. Conclusions and Recommendations

Inspired by the existence of various different apportionment methods in the knowledge area of political voting, this paper has followed a methodology to integrate them with research on *who owns float*, or rather *how should float be optimally apportioned* to critical activities. Combining these considerations, a simulation model has been created. It has tested for which exponent (i.e. proportionality to duration) and with what apportionment method the float pre-allocation will be at an optimum. This paper has also described how to calculate metrics to measure the float overrun performance of many activities in network schedules when an increasing amount of project float is made available. Discrete overruns counts and continuous overrun periods have been represented graphically, which shows at a single glace how critical activities perform. A relatively complex schedule example from the PSPLIB has been tested and the results validate that such a float pre-allocation can be effective and efficient. Several observations can be made:

- It is acknowledged that criticality and float of individual activities likely change during the project execution. Yet the method provides a preallocation (in conjunction with the baseline schedule) that needs not be updated;
- All gains and losses of individual firms who perform critical activities are measured against their preallocated values. If a subcontractor uses less, it could allow successors to use that time in exchange for a fee. If it wants to use more because it encounters delays, it would have to pay such a fee to another subcontractor who is willing to reduce its own allocation by the same amount of time. The value of a day should be contractually established;
- Implementing this approach provides a positive incentive for each firm to perform its work in a timely manner, because it creates a possible bonus-and-cost mechanism within the project itself. Of course, such a 'market of opportunities' needs to be administered in a fair and balanced manner. Project float that remains unused would expire, but may turn be rewarded by the owner with an early completion bonus that can be shared by the firms.

Future research will simulate more complex scheduling phenomena, e.g. a changing critical path due to randomized durations of activities, envision ways to value time in monetary terms, explore the role of the general contractor as a potential administrator of the marketplace, and develop possible contractual provisions to implement it in practice.

Acknowledgement

The support of the National Science Foundation (Grant CMMI-1265989) for portions of the work presented here is gratefully acknowledged. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Balinski, M. L., Young, H. P. (1977). On Huntington methods of apportionment. *Siam Journal on Applied Mathematics*, 33(4): 607-618.

Herroelen, W., Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, 19(5): 559-577.

Jones, M. A., Wilson, J. M. (2010). Evaluation of thresholds for power mean-based and other divisor methods of apportionment. *Mathematical Social Sciences*, 59(3): 343-348.

Kolish, R., Sprecher, A. (1996). PSPLIB – A project scheduling problem library. *European Journal of Operational Research*, 96(1): 205-216.

Stumpf, G. R. (2000). Schedule delay analysis. *Cost Engineering*, 42(7): 32-43.

Tukel, O. I., Rom, W. O., Eksioglu, S. D. (2006). An investigation of buffer sizing techniques in critical chain scheduling. *European Journal of Operational Research*, 172(2): 401-416.

Verba, S. (2003). Would the dream of political equality turn out to be a nightmare? *Perspectives on Politics*, 1(4): 663-679.

Wiseman, J. D. A. (2001). Apportionment, or how to round seat numbers? Personal website, London, Great Britain, ⟨http://www.jdawiseman.com/papers/electsys/apportionment.html⟩, accessed October 31, 2015.

Appendix

Appendix 1: J60_1 Schedule from PSPLIB with Calculated TF, FF, CI

ID	Name	Duration	Distribution	index	Successor	ES	EF	Total Float	Free Float
1	**j1**	**0**	**Fixed**	**N/A**	**j2-FTS-0;j3-FTS-0;j4-FTS-0**	**0**	**0**	**0**	**0**
2	j2	8	Tri	7.2;8;12	j5-FTS-0;j10-FTS-0;j15-FTS-0	0	8	1	0
3	j3	1	Tri	0.9;1;1.5	j7-FTS-0;j14-FTS-0;j29-FTS-0	0	1	18	0
4	**j4**	**10**	**Tri**	**9;10;15**	**j8-FTS-0;j12-FTS-0;j16-FTS-0**	**0**	**10**	**0**	**0**
5	j5	6	Tri	5.4;6;9	j6-FTS-0;j22-FTS-0;j24-FTS-0	8	14	3	0
6	j6	5	Tri	4.5;5;7.5	j17-FTS-0;j38-FTS-0	14	19	4	0
7	j7	8	Tri	7.2;8;12	j23-FTS-0	1	9	18	0
8	**j8**	**9**	**Tri**	**8.1;9;13.5**	**j9-FTS-0;j20-FTS-0;j40-FTS-0**	**10**	**19**	**0**	**0**
9	**j9**	**1**	**Tri**	**0.9;1;1.5**	**j13-FTS-0;j35-FTS-0**	**19**	**20**	**0**	**0**
10	j10	9	Tri	8.1;9;13.5	j11-FTS-0;j45-FTS-0	8	17	1	0
11	j11	8	Tri	7.2;8;12	j26-FTS-0;j37-FTS-0;j44-FTS-0	17	25	1	0
12	j12	3	Tri	2.7;3;4.5	j21-FTS-0;j27-FTS-0	10	13	7	0
13	**j13**	**6**	**Tri**	**5.4;6;9**	**j18-FTS-0**	**20**	**26**	**0**	**0**
14	j14	2	Tri	1.8;2;3	j18-FTS-0;j19-FTS-0;j34-FTS-0	1	3	23	0
15	j15	5	Tri	4.5;5;7.5	j25-FTS-0;j59-FTS-0	8	13	17	0
16	j16	1	Tri	0.9;1;1.5	j55-FTS-0;j58-FTS-0	10	11	47	16
17	j17	3	Tri	2.7;3;4.5	j32-FTS-0;j43-FTS-0	19	22	4	0
18	**j18**	**10**	**Tri**	**9;10;15**	**j28-FTS-0;j33-FTS-0**	**26**	**36**	**0**	**0**
19	j19	9	Tri	8.1;9;13.5	j28-FTS-0	3	12	24	24
20	**j20**	**1**	**Tri**	**0.9;1;1.5**	**j27-FTS-0;j31-FTS-0**	**19**	**20**	**0**	**0**
21	j21	3	Tri	2.7;3;4.5	j39-FTS-0	13	16	26	26
22	j22	6	Tri	5.4;6;9	j31-FTS-0	14	20	3	0
23	j23	3	Tri	2.7;3;4.5	j51-FTS-0	9	12	18	18
24	j24	3	Tri	2.7;3;4.5	j48-FTS-0	14	17	38	29
25	j25	7	Tri	6.3;7;10.5	j40-FTS-0	13	20	17	0
26	j26	6	Tri	5.4;6;9	j49-FTS-0;j54-FTS-0	25	31	1	0
27	**j27**	**10**	**Tri**	**9;10;15**	**j30-FTS-0;j51-FTS-0**	**20**	**30**	**0**	**0**
28	**j28**	**9**	**Tri**	**8.1;9;13.5**	**j47-FTS-0;j61-FTS-0**	**36**	**45**	**0**	**0**

29	j29	8	Tri	7.2;8;12	j41-FTS-0;j57-FTS-0	1	9	22	0
30	j30	4	Tri	3.6;4;6	j56-FTS-0	30	34	13	13
31	j31	3	Tri	2.7;3;4.5	j43-FTS-0	20	23	3	0
32	j32	3	Tri	2.7;3;4.5	j57-FTS-0	22	25	43	43
33	**j33**	**6**	**Tri**	**5.4;6;9**	**j39-FTS-0**	**36**	**42**	**0**	**0**
34	j34	1	Tri	0.9;1;1.5	j44-FTS-0	3	4	38	21
35	j35	9	Tri	8.1;9;13.5	j36-FTS-0;j39-FTS-0	20	29	9	0
36	j36	9	Tri	8.1;9;13.5	j42-FTS-0	29	38	9	0
37	j37	1	Tri	0.9;1;1.5	j58-FTS-0	25	26	38	1
38	j38	2	Tri	1.8;2;3	j50-FTS-0;j60-FTS-0	19	21	13	7
39	**j39**	**4**	**Tri**	**3.6;4;6**	**j53-FTS-0**	**42**	**46**	**0**	**0**
40	j40	9	Tri	8.1;9;13.5	j53-FTS-0	20	29	17	17
41	j41	10	Tri	9;10;15	j46-FTS-0	9	19	22	0
42	j42	8	Tri	7.2;8;12	j48-FTS-0	38	46	9	0
43	j43	4	Tri	3.6;4;6	j51-FTS-0;j58-FTS-0	23	27	3	0
44	j44	3	Tri	2.7;3;4.5	j50-FTS-0	25	28	17	0
45	j45	6	Tri	5.4;6;9	j53-FTS-0	17	23	23	23
46	j46	6	Tri	5.4;6;9	j56-FTS-0	19	25	22	22
47	**j47**	**7**	**Tri**	**6.3;7;10.5**	**j52-FTS-0**	**45**	**52**	**0**	**0**
48	j48	3	Tri	2.7;3;4.5	j55-FTS-0	46	49	9	9
49	j49	2	Tri	1.8;2;3	j60-FTS-0	31	33	1	1
50	j50	10	Tri	9;10;15	j61-FTS-0	28	38	17	17
51	**j51**	**4**	**Tri**	**3.6;4;6**	**j60-FTS-0**	**30**	**34**	**0**	**0**
52	**j52**	**2**	**Tri**	**1.8;2;3**	**j54-FTS-0**	**52**	**54**	**0**	**0**
53	**j53**	**1**	**Tri**	**0.9;1;1.5**	**j56-FTS-0**	**46**	**47**	**0**	**0**
54	**j54**	**4**	**Tri**	**3.6;4;6**	**j55-FTS-0**	**54**	**58**	**0**	**0**
55	**j55**	**10**	**Tri**	**9;10;15**	**j57-FTS-0**	**58**	**68**	**0**	**0**
56	**j56**	**8**	**Tri**	**7.2;8;12**	**j61-FTS-0**	**47**	**55**	**0**	**0**
57	**j57**	**6**	**Tri**	**5.4;6;9**	**j59-FTS-0**	**68**	**74**	**0**	**0**
58	j58	10	Tri	9;10;15	j59-FTS-0	27	37	37	37
59	**j59**	**3**	**Tri**	**2.7;3;4.5**	**j62-FTS-0**	**74**	**77**	**0**	**0**
60	**j60**	**10**	**Tri**	**9;10;15**	**j62-FTS-0**	**34**	**44**	**0**	**0**
61	**j61**	**1**	**Tri**	**0.9;1;1.5**	**j62-FTS-0**	**55**	**56**	**0**	**0**
62	**j62**	**0**	**Fixed**	**N/A**	**N/A**	**77**	**77**	**0**	**0**