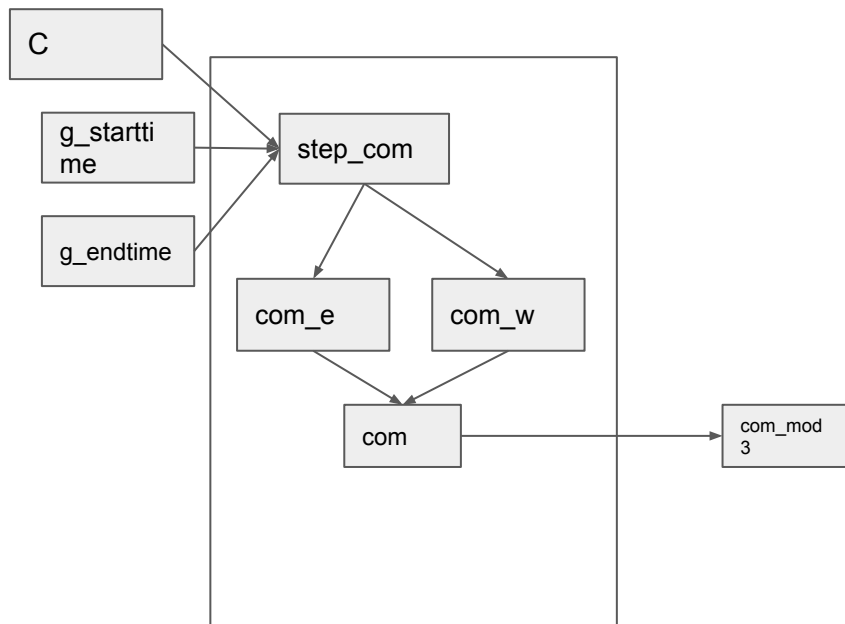


Modules and their relationship

Bilal

datawiz

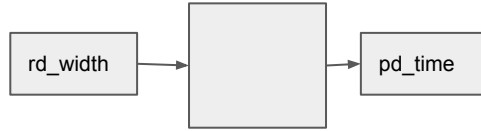
2.combinations



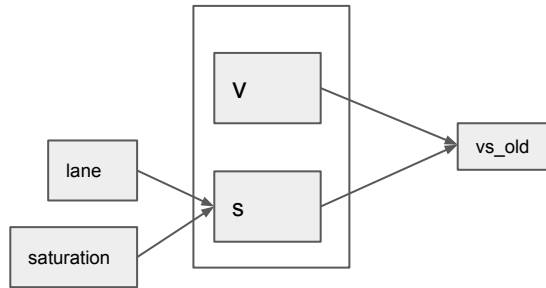
```

1 #start time of the green time
2 g_starttime<- c(0,0,0,0) #random choosing to match the combinations using int1 to 4
3 #end time of green time of east bound # for overlapping phases we will add the phase for that movement
4
5 g_endtime1<- g_starttime[1]+phasing:phase2[1]
6 g_endtime2<- g_starttime[2]+phasing:phase2[2]
7 g_endtime3<- g_starttime[3]+phasing:phase1[3]+phasing:phase2[3]
8 g_endtime33<- g_starttime[3]+phasing:phase2[3]
9 g_endtime4<- g_starttime[4]+phasing:phase2[4]
10 g_endtime44<- g_starttime[4]+phasing:phase1[4]+phasing:phase2[4]
11 g_endtime_e<- c(g_endtime1,g_endtime2,g_endtime3,g_endtime4)
12 g_endtime_w<- c(g_endtime1,g_endtime2,g_endtime33,g_endtime44)
13
14 |-----|
15 #combinations of all green split movements over the cycle length.
16 step_com<- function(C, g_starttime, g_endtime){
17
18   step<- C*(g_endtime-g_starttime)
19   step1<- as.vector(1:step[1])
20   step2<- as.vector(1:step[2])
21   step3<- as.vector(1:step[3])
22   step4<- as.vector(1:step[4])
23   step_com<- expand.grid(step1,step2,step3,step4)
24   return(step_com)
25 }
26
27 com_e<- step_com(C, g_starttime, g_endtime_e)
28 com_w<- step_com(C, g_starttime, g_endtime_w)
29 #to minimize the total number of iterations
30 com<- com_e[!(com_e$Var4>max(com_w$Var4)),]
31 com<- com[1:10,]
32 # trying to reduce the number of combinations effort
33 com_mod<- com
34 com_mod$diff12<- com_mod$Var1 - com_mod$Var2
35 com_mod$diff13<- com_mod$Var1 - com_mod$Var3
36 com_mod$diff14<- com_mod$Var1 - com_mod$Var4
37 com_mod2<- com_mod[duplicated(com_mod[,5:7]),]
38 com_mod3<- com_mod2[,c(1:4)]
39 com_mod3<- com_mod3[c(1:10),]
  
```

3. Pedestrian time



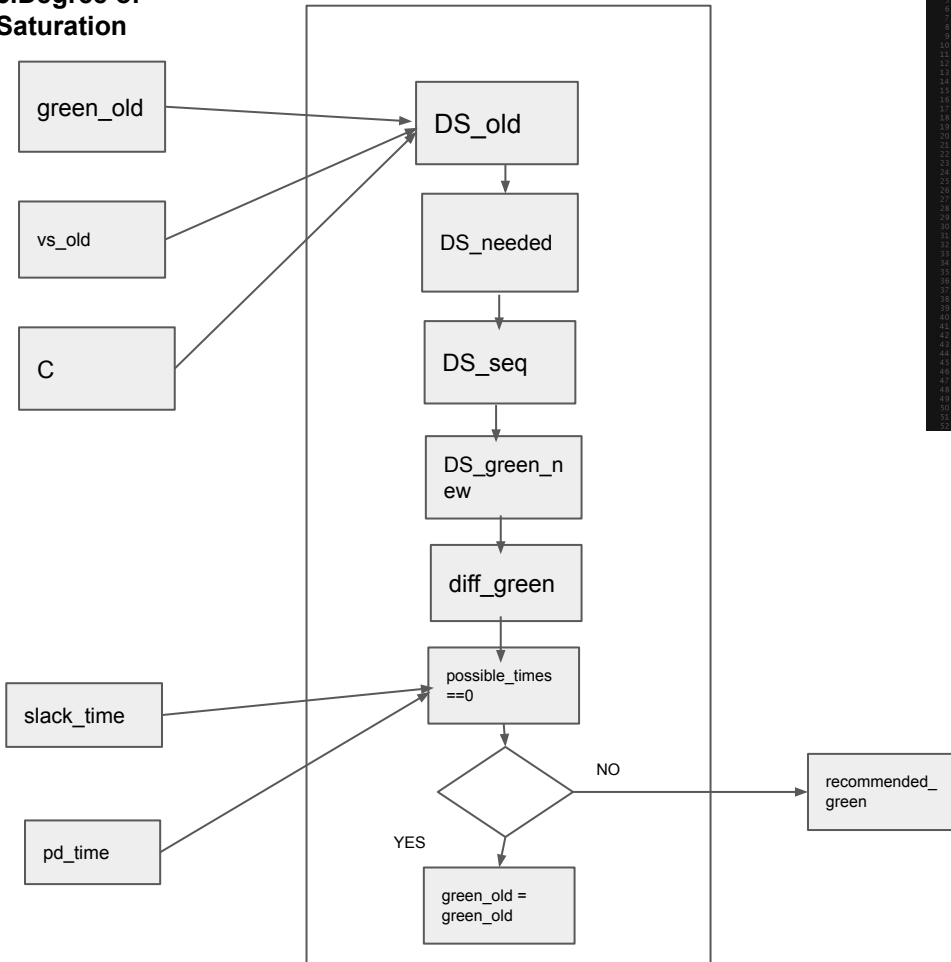
1. V/S ratio



```

25 #selected phasing scheme and its corresponding slack time
26 #check if the difference of new and old green lies inside the slack time obtained
27 #if not, take another phasing scheme whose difference lies inside the slack time selected
28 #if yes, change the green time of the suggested phasing to new green and calculate again to find another optimized phasing scheme
29 intersection_num<- 1
30 ds_phasing_standard<- 2
31
32
33 #existing phasing
34 phasing$phase1m<- c("3", "7", "8,3", "4,7")
35 phasing$phase2m<- c("8,4", "8,4", "8,4", "8,4")
36 phasing$phase3m<- c("5", "1", "7", "1")
37 phasing$phase4m<- c("0", "0", "0", "2,5")
38 phasing$phase5m<- c("0", "0", "0", "1,6")
39
40 #volume
41 phasing$ET<- c(2820,2616,1730,2495)
42 phasing$EL<- c(399,0,560,0)
43 phasing$WT<- c(1383,1372,1006,1878)
44 phasing$WL<- c(0,64,91,156)
45 phasing$NT<- c(0,0,333,0)
46 phasing$NL<- c(0,193,130,126)
47 phasing$ST<- c(0,0,660,0)
48 phasing$SL<- c(137,0,979,0)
49
50 #saturation per lane and total lanes
51 saturation<- data.frame("ET"=rep(2000,4),"WT"=rep(2000,4),"NT"=rep(2000,4),"ST"=rep(2000,4),
52 "EL"=rep(1800,4),"WL"=rep(1800,4),"NL"=rep(1800,4),"SL"=rep(1800,4),
53 "lane_ET" = rep(4,4),"lane_WT" = rep(4,4),"lane_NT" = c(1,2,1,1),"lane_ST" = c(1,0,4,0),
54 "lane_EL" = rep(1,4),"lane_WL" = rep(1,4),"lane_NL" = rep(1,4),"lane_SL" = rep(1,4))
55
56 saturation$ET_s<- (saturation$ET*(saturation$lane_ET))
57 saturation$WT_s<- (saturation$WT*(saturation$lane_WT))
58 saturation$NT_s<- (saturation$NT*(saturation$lane_NT))
59 saturation$ST_s<- (saturation$ST*(saturation$lane_ST))
60 saturation$EL_s<- (saturation$EL*(saturation$lane_EL))
61 saturation$WL_s<- (saturation$WL*(saturation$lane_WL))
62 saturation$NL_s<- (saturation$NL*(saturation$lane_NL))
63 saturation$SL_s<- (saturation$SL*(saturation$lane_SL))
64
65 #V/S
66 vs_old<- data.frame("ET_V/s"=(phasing$ET/saturation$ET_s),"WT_V/s"=(phasing$WT/saturation$WT_s),
67 "NT_V/s"=(phasing$NT/saturation$NT_s),"ST_V/s"=(phasing$ST/saturation$ST_s),
68 "EL_V/s"=(phasing$EL/saturation$EL_s),"WL_V/s"=(phasing$WL/saturation$WL_s),
69 "NL_V/s"=(phasing$NL/saturation$NL_s),"SL_V/s"=(phasing$SL/saturation$SL_s))
  
```

5.Degree of Saturation

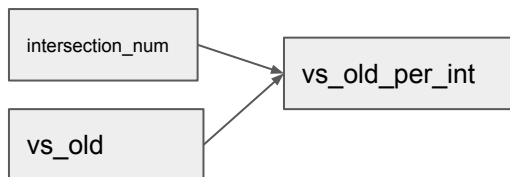


Variable definitions

[illegible][illegible]

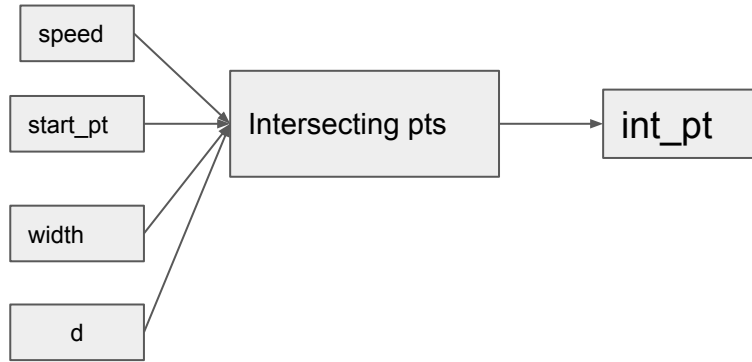
Variable definitions

4.V/s for each intersection



```
64  
65 #V/S  
66 vs_old<- data.frame("ET_v/s"=(phasing$ET/saturation$ET_s),"WT_v/s"=(phasing$WT/saturation$WT_s),  
67 "NT_v/s"=(phasing$NT/saturation$NT_s),"ST_v/s"=(phasing$ST/saturation$ST_s),  
68 "EL_v/s"=(phasing$EL/saturation$EL_s),"WL_v/s"=(phasing$WL/saturation$WL_s),  
69 "NL_v/s"=(phasing$NL/saturation$NL_s),"SL_v/s"=(phasing$SL/saturation$SL_s))  
70 vs_old[is.na(vs_old)] = 0  
71 vs_old_vector<- as.matrix(vs_old[intersection_num,])#change to whichever intersection you want  
72 vs_old_vector<- as.vector(vs_old_vector)  
73 vs_old_movs<- c(8,4,6,2,3,7,1,5)  
74 vs_old_per_int<- cbind.data.frame(vs_old_vector,vs_old_movs)  
75
```

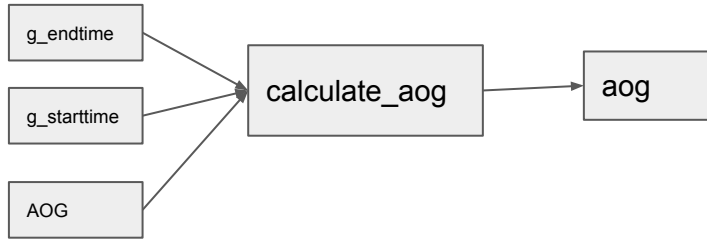
6. Point of intersection function



Variable definitions

```
92  
93 #function to find the intersection pts.....  
94 intersect_pts <- function(speed,start_pt,width,dist) {  
95   A<- matrix(c(speed,1,1,0), nrow = 2, ncol = 2, byrow = TRUE)  
96   B<- matrix(c(start_pt+width,dist), nrow = 2, ncol = 1, byrow = TRUE)  
97   invA<- solve(A)  
98   int_pt<- invA %*% B  
99   int_pt<- as.vector(int_pt)  
100   return(int_pt)  
101 }  
102
```

7.AOG calculation function



Variable
definitions

```
103 #function to find the location of max AOG position and number of veh passing.....
104 calculate_aog<- function(g_endtime,g_starttime,AOG){
105   vector10<-vector(mode = "numeric")
106   # c=1
107   while(!g_endtime>C-1){
108     g_starttime<- g_starttime+1
109     g_endtime<- g_endtime+1
110     count_before<- sum(AOG > g_starttime & AOG < g_endtime)
111     vector10<- append(vector10,count_before)
112   }
113   # b_count<- list("place" = which.max(vector10), "Arrival_num" = max(vector10))
114   return(vector10)
115 }
116
117
```

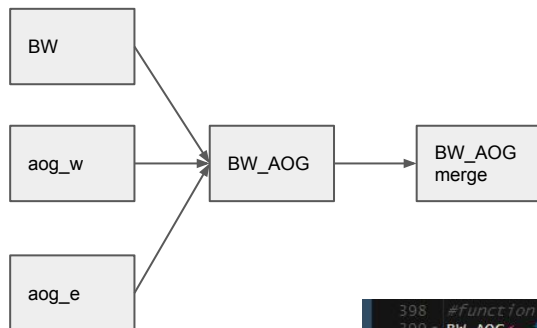

9.Start and end time of green

g_starttime

g_endtime

```
50 #start time of the green time
51 g_starttime<- c(0,0,0,0)#random choosing to match the combinations using inti to 4
52 #end time of green time of east bound # for overlapping phases we will add the phase for that movement
53 g_endtime1<- g_starttime[1]+phasing$phase2[1]
54 g_endtime2<- g_starttime[2]+phasing$phase2[2]
55 g_endtime3<- g_starttime[3]+phasing$phase1[3]+phasing$phase2[3]
56 g_endtime33<-g_starttime[3]+phasing$phase2[3]
57 g_endtime4<- g_starttime[4]+phasing$phase2[4]
58 g_endtime44<-g_starttime[4]+phasing$phase1[4]+phasing$phase2[4]
59 g_endtime_e<- c(g_endtime1,g_endtime2,g_endtime3,g_endtime4)
60 g_endtime_w<- c(g_endtime1,g_endtime2,g_endtime33,g_endtime44)
61
```

10. Merging AOG and BW



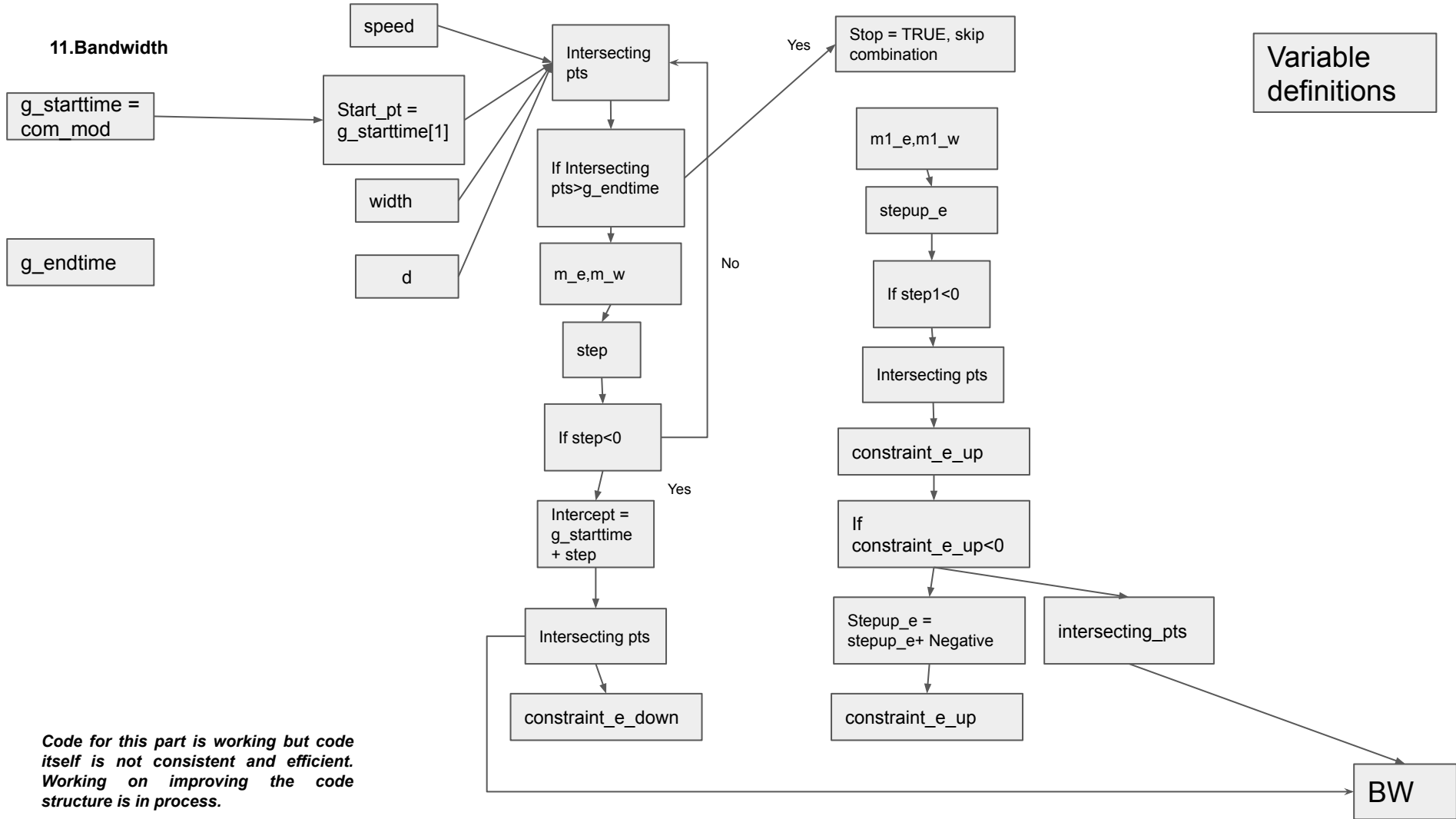
Variable definitions

```

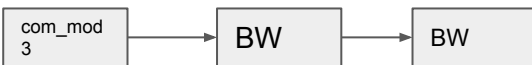
398 #function to combine BW and AOG calculated
399 BW_AOG<- function(BW,aog_e, aog_w){
400   aog_e1<- as.data.frame(unlist(aog_e[[1]]))
401   aog_e2<- as.data.frame(unlist(aog_e[[2]]))
402   aog_e3<- as.data.frame(unlist(aog_e[[3]]))
403   aog_e4<- as.data.frame(unlist(aog_e[[4]]))
404
405   aog_e1$com1<- 1:length(aog_e[[1]])
406   aog_e2$com2<- 1:length(aog_e[[2]])
407   aog_e3$com3<- 1:length(aog_e[[3]])
408   aog_e4$com4<- 1:length(aog_e[[4]])
409
410   BW_AOGmerge<- merge(x = BW, y = aog_e1, by = "com1", all.x = TRUE)
411   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_e2, by = "com2", all.x = TRUE)
412   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_e3, by = "com3", all.x = TRUE)
413   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_e4, by = "com4", all.x = TRUE)
414
415   aog_w1<- as.data.frame(unlist(aog_w[[1]]))
416   aog_w2<- as.data.frame(unlist(aog_w[[2]]))
417   aog_w3<- as.data.frame(unlist(aog_w[[3]]))
418   aog_w4<- as.data.frame(unlist(aog_w[[4]]))
419
420   aog_w1$com1<- 1:length(aog_w[[1]])
421   aog_w2$com2<- 1:length(aog_w[[2]])
422   aog_w3$com3<- 1:length(aog_w[[3]])
423   aog_w4$com4<- 1:length(aog_w[[4]])
424
425   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_w1, by = "com1", all.x = TRUE)
426   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_w2, by = "com2", all.x = TRUE)
427   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_w3, by = "com3", all.x = TRUE)
428   BW_AOGmerge<- merge(x = BW_AOGmerge, y = aog_w4, by = "com4", all.x = TRUE)
429   BW_AOGmerge[, 5:14] <- sapply(BW_AOGmerge[, 5:14], as.numeric)
430   return(BW_AOGmerge)
431 }
432

```

11.Bandwidth



8.Band Width function



Variable definitions

```

110 function to find bus
111 BW= function(com)
112
113 my_vector_e1 <- vector(mode="numeric")
114 my_vector_e2 <- vector(mode="numeric")
115 my_vector_e3 <- vector(mode="numeric")
116 my_vector_e4 <- vector(mode="numeric")
117
118 # my_vector_w <- vector(mode="numeric")
119
120
121 my_vector_w1 <- vector(mode="numeric")
122 my_vector_w2 <- vector(mode="numeric")
123 my_vector_w3 <- vector(mode="numeric")
124 my_vector_w4 <- vector(mode="numeric")
125
126 count_step <- count_step1
127
128 for (i in 1:nrow(com)){
129   g_starttime <- com[i,]
130   g_endtime1 <- g_starttime[1]+phasingphase2[1]
131   g_endtime2 <- g_starttime[1]+phasingphase2[2]
132   g_endtime3 <- g_starttime[1]+phasingphase2[3]
133   g_endtime4 <- g_starttime[1]+phasingphase2[4]
134   g_endtime_w1 <- g_starttime[1]+phasingphase2[1]
135   g_endtime_w2 <- g_starttime[1]+phasingphase2[2]
136   g_endtime_w3 <- g_starttime[1]+phasingphase2[3]
137   g_endtime_w4 <- g_starttime[1]+phasingphase2[4]
138
139   # defining bottom time .....
140   # defining the intersection data for 4 bands
141   inpt_e <- NULL
142   inpt_w <- NULL
143   stop1 <- FALSE
144   stop4 <- FALSE
145   start_pt_w <- g_starttime[4]+(speed_w*d[4])
146   stop1 <- FALSE
147   stop4 <- FALSE
148   stop_new <- FALSE
149
150   for (i in 1:4){
151     inpt_e[i] <- intersect_pts(speed_e,as.numeric(g_starttime[i]),0,d[i])[2]
152
153     if (inpt_e[i] < g_endtime_e[i]){
154       inpt_e <- NULL
155       stop1 <- TRUE
156       break
157     }
158   }
159 }

```

```

159 for (i in 1:4){
160   inpt_e[i] <- intersect_pts(speed_e,as.numeric(intercept_e),0,d[i])[2]
161 }
162
163 if (inpt_e[1] < g_endtime_e[1]){
164   inpt_e <- NULL
165   intercept_e <- g_starttime[1]
166   stop4 <- TRUE
167   break
168 }
169
170 try{
171   (stop4)
172   } silent = TRUE
173 }
174
175 #bottom time drum
176 constrain_e_down <- inpt_e+g_starttime
177
178 # defining the intersection .....
179 if (stop1){
180   inpt_w <- inpt_w
181   constrain_w_down <- inpt_w+g_starttime
182 }
183
184 if (stop1){
185   intercept_w <- start_pt_w+stop1
186 }
187
188 for (i in 1:4){
189   inpt_w[i] <- intersect_pts(speed_w,as.numeric(intercept_w),0,d[i])[2]
190 }
191
192 if (inpt_w[1] < g_endtime_w[1]){
193   inpt_w <- NULL
194   intercept_w <- start_pt_w
195   stop5 <- TRUE
196   break
197 }
198
199 try{
200   (stop5)
201   } silent = TRUE
202 }
203
204 #bottom time drum
205 constrain_w_down <- inpt_w+g_starttime

```

```

206 constrain_e_up <- as.numeric(g_endtime_e) as.numeric(inpt_e+up)
207 width_e <- stop4
208
209 if (any(is.negative(constraint_e_up) ~ FALSE)){
210   negative <- min(which(constraint_e_up < 0)) changed from max to min
211   negative <- min(which(constraint_e_up < 0))
212   stop4 <- stop4+negative
213   all_line <- intersect_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
214   d[i] = i+1
215   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
216   # the number of points before and of green split
217 }
218
219 constrain_w_up <- as.numeric(g_endtime_w) as.numeric(inpt_w+up)
220 width_w <- stop5
221
222 if (any(is.negative(constraint_w_up) ~ FALSE)){
223   negative <- min(which(constraint_w_up < 0))
224   negative <- min(which(constraint_w_up < 0))
225   stop5 <- stop5+negative
226   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
227   d[i] = i+1
228   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
229   # the number of points before and of green split
230 }
231
232 constrain_e_up <- as.numeric(g_endtime_e) as.numeric(inpt_e+up)
233 width_e <- stop4
234
235 if (any(is.negative(constraint_e_up) ~ FALSE)){
236   negative <- min(which(constraint_e_up < 0))
237   negative <- min(which(constraint_e_up < 0))
238   stop4 <- stop4+negative
239   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
240   d[i] = i+1
241   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
242   # the number of points before and of green split
243 }
244
245 constrain_w_up <- as.numeric(g_endtime_w) as.numeric(inpt_w+up)
246 width_w <- stop5
247
248 if (any(is.negative(constraint_w_up) ~ FALSE)){
249   negative <- min(which(constraint_w_up < 0))
250   negative <- min(which(constraint_w_up < 0))
251   stop5 <- stop5+negative
252   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
253   d[i] = i+1
254   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
255   # the number of points before and of green split
256 }
257
258 my_vector_e1 <- append(my_vector_e1,width_e)
259 my_vector_e2 <- append(my_vector_e2,width_e)
260 my_vector_e3 <- append(my_vector_e3,width_e)
261 my_vector_e4 <- append(my_vector_e4,width_e)
262 my_vector_w1 <- append(my_vector_w1,width_w)
263 my_vector_w2 <- append(my_vector_w2,width_w)
264 my_vector_w3 <- append(my_vector_w3,width_w)
265 my_vector_w4 <- append(my_vector_w4,width_w)
266
267 count_step <- count_step
268
269 my_vector_e1 <- append(my_vector_e1,width_e)
270 my_vector_e2 <- append(my_vector_e2,width_e)
271 my_vector_e3 <- append(my_vector_e3,width_e)
272 my_vector_e4 <- append(my_vector_e4,width_e)
273 my_vector_w1 <- append(my_vector_w1,width_w)
274 my_vector_w2 <- append(my_vector_w2,width_w)
275 my_vector_w3 <- append(my_vector_w3,width_w)
276 my_vector_w4 <- append(my_vector_w4,width_w)
277
278 print(count_step)

```

```

160 try{
161   (stop1)
162   } silent = TRUE
163 }
164
165 for (i in 1:4){
166   inpt_e[i] <- intersect_pts(speed_e,as.numeric(start_pt_w),0,d[i])[2]
167 }
168
169 if (inpt_e[1] < g_endtime_e[1]){
170   inpt_e <- NULL
171   stop1 <- TRUE
172   break
173 }
174
175 try{
176   (stop1)
177   } silent = TRUE
178 }
179
180 # defining the intersection .....
181 # defining the intersection data for 4 bands
182 inpt_e <- NULL
183 inpt_w <- NULL
184 stop1 <- FALSE
185 stop4 <- FALSE
186 start_pt_w <- g_starttime[4]+(speed_w*d[4])
187 stop1 <- FALSE
188 stop4 <- FALSE
189 stop_new <- FALSE
190
191 for (i in 1:4){
192   inpt_e[i] <- intersect_pts(speed_e,as.numeric(g_starttime[i]),0,d[i])[2]
193 }
194
195 if (inpt_e[1] < g_endtime_e[1]){
196   inpt_e <- NULL
197   stop1 <- TRUE
198   break
199 }
200
201 try{
202   (stop1)
203   } silent = TRUE
204 }
205
206 #bottom time drum
207 constrain_e_down <- inpt_e+g_starttime
208
209 # defining the intersection .....
210 if (stop1){
211   inpt_w <- inpt_w
212   constrain_w_down <- inpt_w+g_starttime
213 }
214
215 if (stop1){
216   intercept_w <- start_pt_w+stop1
217 }
218
219 for (i in 1:4){
220   inpt_w[i] <- intersect_pts(speed_w,as.numeric(intercept_w),0,d[i])[2]
221 }
222
223 if (inpt_w[1] < g_endtime_w[1]){
224   inpt_w <- NULL
225   intercept_w <- start_pt_w
226   stop5 <- TRUE
227   break
228 }
229
230 try{
231   (stop5)
232   } silent = TRUE
233 }
234
235 #bottom time drum
236 constrain_w_down <- inpt_w+g_starttime

```

```

237 my_vector_e1 <- append(my_vector_e1,width_e)
238 my_vector_e2 <- append(my_vector_e2,width_e)
239 my_vector_e3 <- append(my_vector_e3,width_e)
240 my_vector_e4 <- append(my_vector_e4,width_e)
241 my_vector_w1 <- append(my_vector_w1,width_w)
242 my_vector_w2 <- append(my_vector_w2,width_w)
243 my_vector_w3 <- append(my_vector_w3,width_w)
244 my_vector_w4 <- append(my_vector_w4,width_w)
245
246 count_step <- count_step
247
248 my_vector_e1 <- append(my_vector_e1,width_e)
249 my_vector_e2 <- append(my_vector_e2,width_e)
250 my_vector_e3 <- append(my_vector_e3,width_e)
251 my_vector_e4 <- append(my_vector_e4,width_e)
252 my_vector_w1 <- append(my_vector_w1,width_w)
253 my_vector_w2 <- append(my_vector_w2,width_w)
254 my_vector_w3 <- append(my_vector_w3,width_w)
255 my_vector_w4 <- append(my_vector_w4,width_w)
256
257 print(count_step)

```

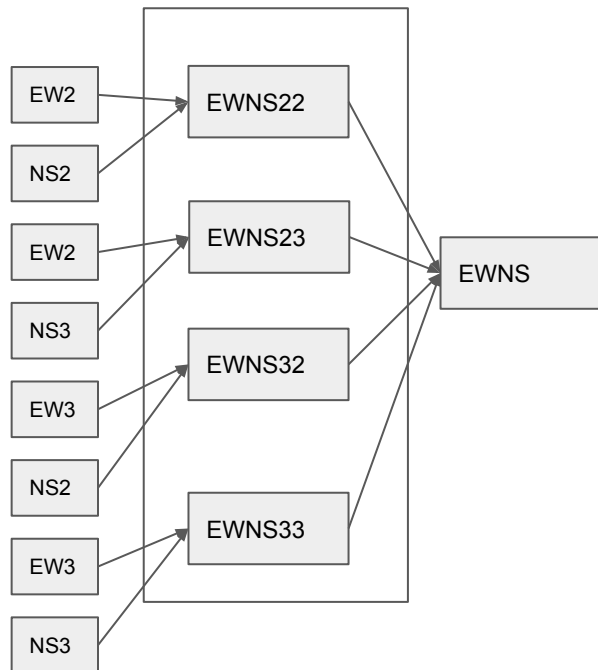
```

258 constrain_e_up <- as.numeric(g_endtime_e) as.numeric(inpt_e+up)
259 width_e <- stop4
260
261 if (any(is.negative(constraint_e_up) ~ FALSE)){
262   negative <- min(which(constraint_e_up < 0))
263   negative <- min(which(constraint_e_up < 0))
264   stop4 <- stop4+negative
265   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
266   d[i] = i+1
267   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
268   # the number of points before and of green split
269 }
270
271 constrain_w_up <- as.numeric(g_endtime_w) as.numeric(inpt_w+up)
272 width_w <- stop5
273
274 if (any(is.negative(constraint_w_up) ~ FALSE)){
275   negative <- min(which(constraint_w_up < 0))
276   negative <- min(which(constraint_w_up < 0))
277   stop5 <- stop5+negative
278   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
279   d[i] = i+1
280   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
281   # the number of points before and of green split
282 }
283
284 constrain_e_up <- as.numeric(g_endtime_e) as.numeric(inpt_e+up)
285 width_e <- stop4
286
287 if (any(is.negative(constraint_e_up) ~ FALSE)){
288   negative <- min(which(constraint_e_up < 0))
289   negative <- min(which(constraint_e_up < 0))
290   stop4 <- stop4+negative
291   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
292   d[i] = i+1
293   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
294   # the number of points before and of green split
295 }
296
297 constrain_w_up <- as.numeric(g_endtime_w) as.numeric(inpt_w+up)
298 width_w <- stop5
299
300 if (any(is.negative(constraint_w_up) ~ FALSE)){
301   negative <- min(which(constraint_w_up < 0))
302   negative <- min(which(constraint_w_up < 0))
303   stop5 <- stop5+negative
304   all_line <- intercept_w_stepup_w, b=speed_w, h=NULL, v=NULL, col="red", led = 2)
305   d[i] = i+1
306   inpt_w[i+1] <- intersect_pts(speed_w,as.numeric(intercept_w+stop4_w,d[i])[2]
307   # the number of points before and of green split
308 }
309
310 my_vector_e1 <- append(my_vector_e1,width_e)
311 my_vector_e2 <- append(my_vector_e2,width_e)
312 my_vector_e3 <- append(my_vector_e3,width_e)
313 my_vector_e4 <- append(my_vector_e4,width_e)
314 my_vector_w1 <- append(my_vector_w1,width_w)
315 my_vector_w2 <- append(my_vector_w2,width_w)
316 my_vector_w3 <- append(my_vector_w3,width_w)
317 my_vector_w4 <- append(my_vector_w4,width_w)
318
319 count_step <- count_step
320
321 my_vector_e1 <- append(my_vector_e1,width_e)
322 my_vector_e2 <- append(my_vector_e2,width_e)
323 my_vector_e3 <- append(my_vector_e3,width_e)
324 my_vector_e4 <- append(my_vector_e4,width_e)
325 my_vector_w1 <- append(my_vector_w1,width_w)
326 my_vector_w2 <- append(my_vector_w2,width_w)
327 my_vector_w3 <- append(my_vector_w3,width_w)
328 my_vector_w4 <- append(my_vector_w4,width_w)
329
330 print(count_step)

```

12. Phasing

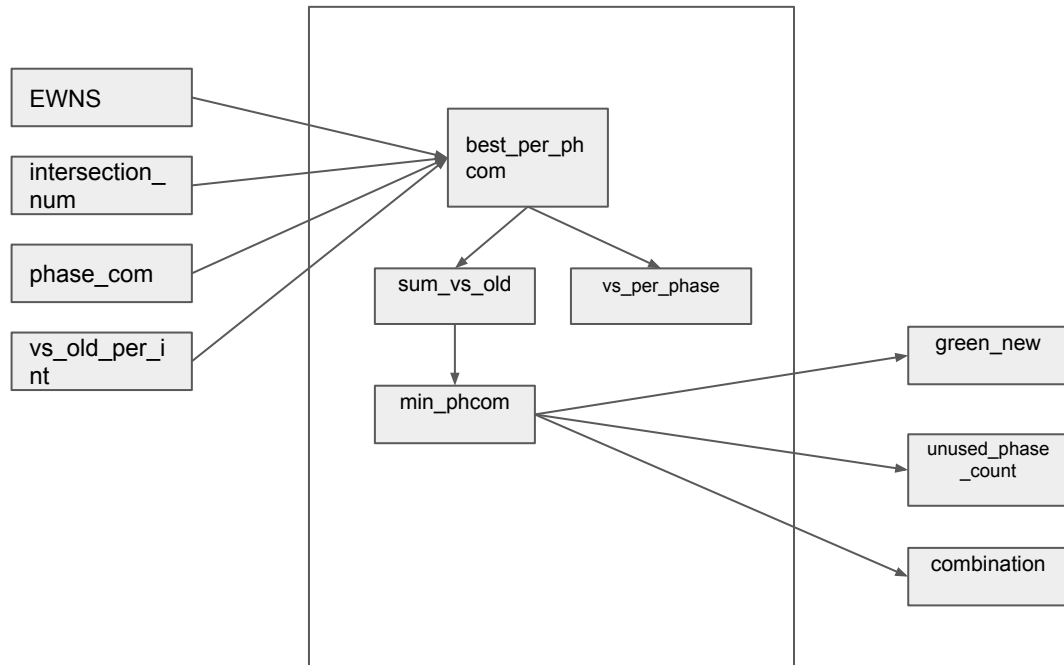
Variable
definitions



```
76 #more than two movements not possible per phase as they are conflicting
77 # m_2ph<- as.data.frame(t(combn(int1,2)))
78 # m_2ph<- m_2ph[-c(1,6),]
79 # m1<- paste(m_2ph$v1,"-",m_2ph$v2)
80 # m2<- as.data.frame(t(combn(m1,2)))
81 # m3<- as.data.frame(t(combn(m1,3)))
82
83 EW2<- data.frame("v1"=c(3),"v2"=c(7),"v3"=c(4),"v4"=c(8))
84 NS2<- data.frame("v1"=c(1,1),"v2"=c(5,6),"v3"=c(2,2),"v4"=c(6,5))
85 EW3<- data.frame("v1"=c(3,3),"v2"=c(7,7),"v3"=c(4,3),"v4"=c(7,8),"v5"=c(4,4),"v6"=c(8,8))
86 NS3<- data.frame("v1"=c(1,1),"v2"=c(5,5),"v3"=c(2,1),"v4"=c(5,6),"v5"=c(2,2),"v6"=c(6,6))
87
88 EW2_NS2<-cbind(EW2,NS2)
89 EWNS_22<- EW2_NS2
90
91 EW2_NS3<- cbind(EW2,NS3)
92 EWNS_23<- EW2_NS3
93
94 EW3_NS2<- cbind(rbind(EW3,EW3[nrow(EW3):1,]),NS2)
95 EWNS_32<- EW3_NS2
96
97 EW3_NS3<- cbind(rbind(EW3,EW3[nrow(EW3):1,]),NS3)
98 EWNS_33<- EW3_NS3
99
100 #take values of vs and put them accordingly
101 sum_vs_old <- vector(mode="numeric")
102 vs_per_phase <- vector(mode="numeric")
103
```

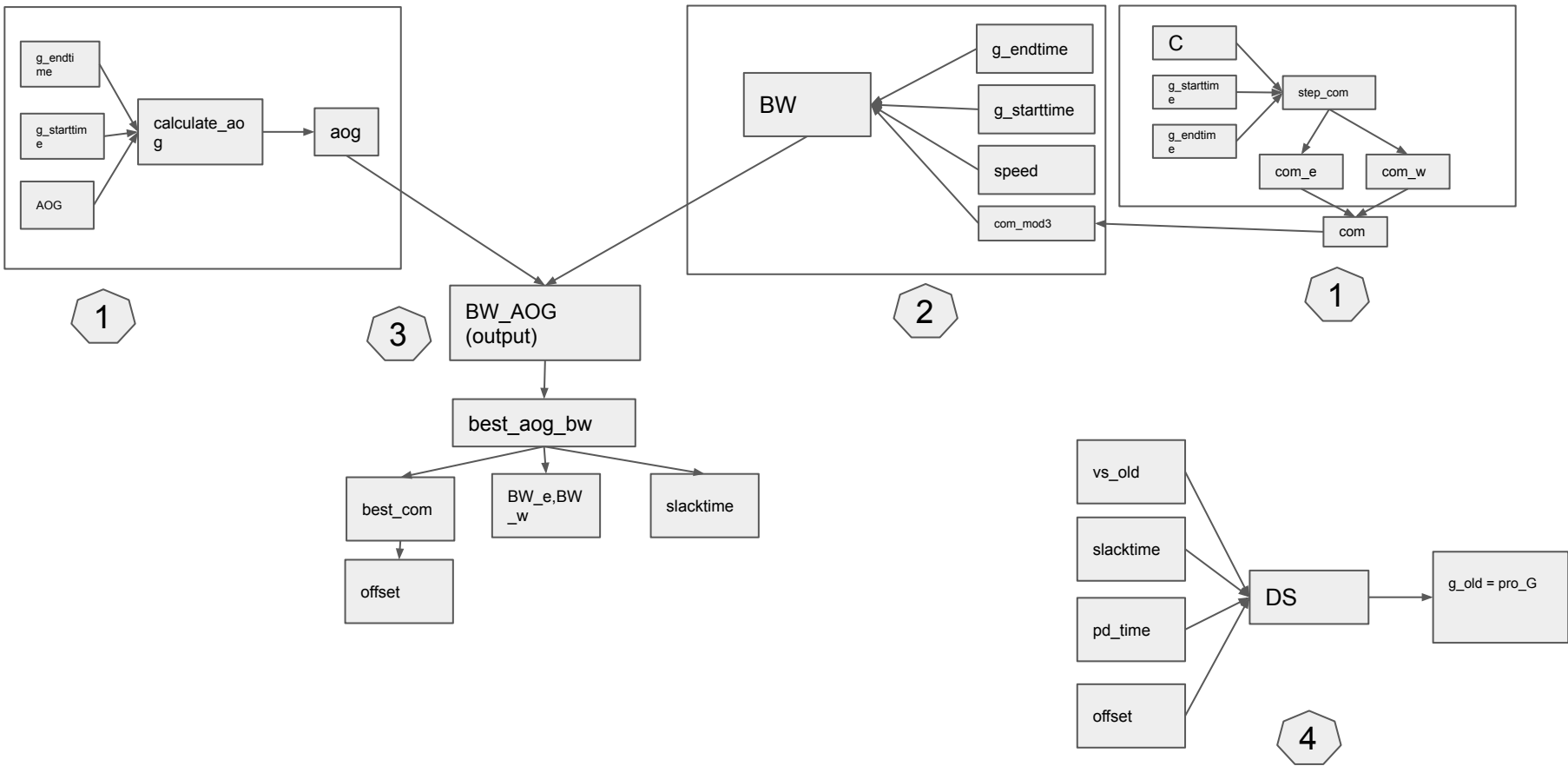
12. Phasing

Variable definitions



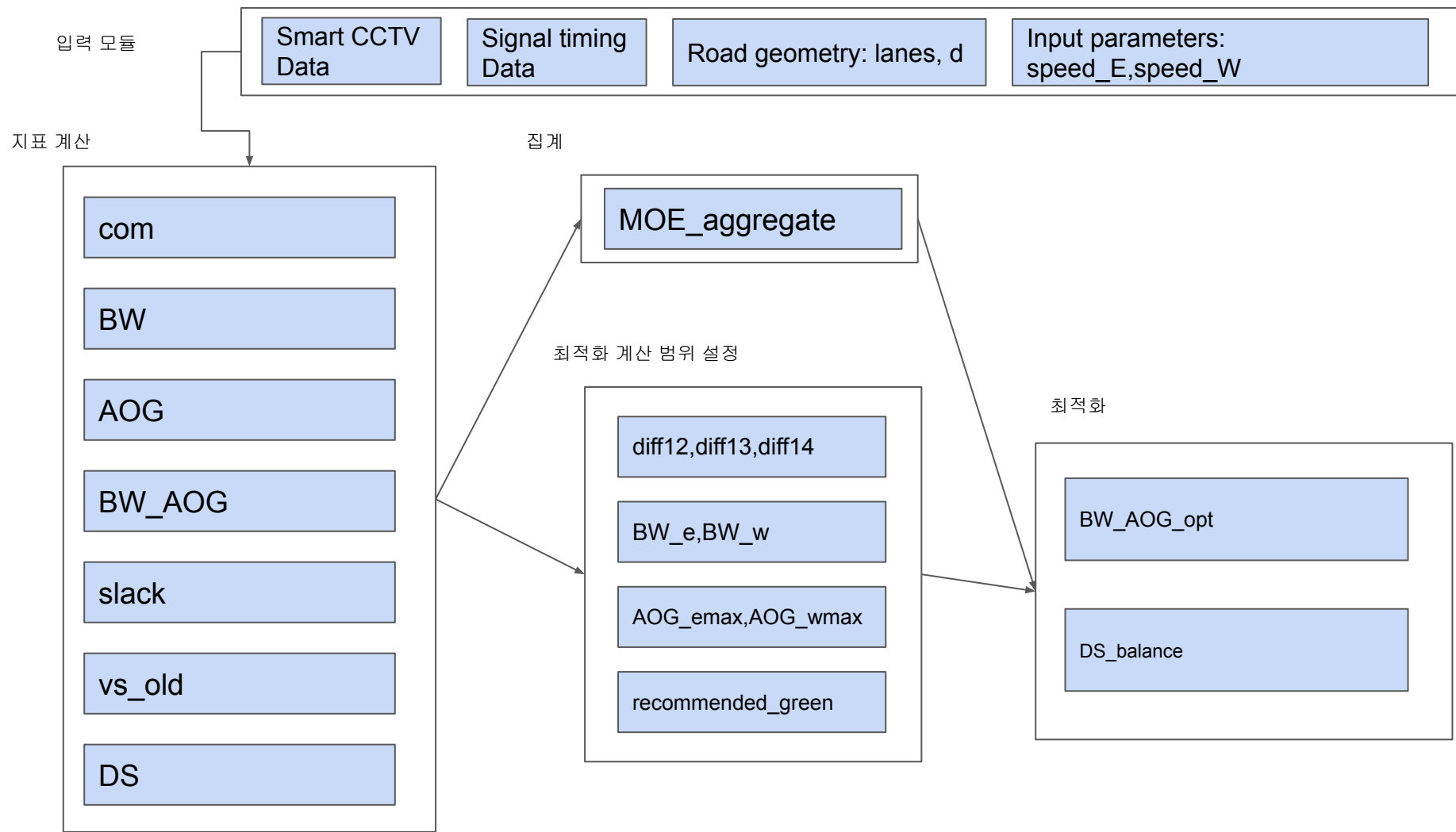
```

105 best_per_phcom<- function(EWNS,vs_old_per_int,phase_com,intersection_num){
106
107
108   for (i in 1:nrow(EWNS)){
109
110     xxxxx<- EWNS[i,]
111     xxxxx<- as.data.frame(as.vector(as.matrix(xxxxx)))
112     xxxxx$id<- 1:nrow(xxxxx)
113     yyy<- merge(xxxx,vs_old_per_int, all = TRUE, by.x = "as.vector(as.matrix(xxxxx))", by.y = "vs_old_mvs")
114     yyy<- yyy[order(yyy$id),]
115
116     if(phase_com == 22){
117       original_phase_count<- 4 #needs to be calculated
118       ph1<- max(yyy[c(T,F,F,F,F,F,F,F),3],yyy[c(F,T,F,F,F,F,F,F),3])
119       ph2<- max(yyy[c(F,F,T,F,F,F,F,F),3],yyy[c(F,F,T,F,F,F,F,F),3])
120       ph3<- max(yyy[c(F,F,F,T,F,F,F,F),3],yyy[c(F,F,F,T,F,F,F,F),3])
121       ph4<- max(yyy[c(F,F,F,F,T,F,F,F),3],yyy[c(F,F,F,F,T,F,F,F),3])
122       ph<- c(ph1,ph2,ph3,ph4)
123       vs_sum<- sum(ph)
124     }else if (phase_com == 23){
125       original_phase_count<- 5
126       ph1<- max(yyy[c(T,F,F,F,F,F,F,F,F),3],yyy[c(F,T,F,F,F,F,F,F,F),3])
127       ph2<- max(yyy[c(F,T,F,F,F,F,F,F,F),3],yyy[c(F,F,T,F,F,F,F,F,F),3])
128       ph3<- max(yyy[c(F,F,F,T,F,F,F,F,F),3],yyy[c(F,F,F,T,F,F,F,F,F),3])
129       ph4<- max(yyy[c(F,F,F,F,T,F,F,F,F),3],yyy[c(F,F,F,F,T,F,F,F,F),3])
130       ph5<- max(yyy[c(F,F,F,F,F,T,F,F,F),3],yyy[c(F,F,F,F,F,T,F,F,F),3])
131       ph<- c(ph1,ph2,ph3,ph4,ph5)
132       vs_sum<- sum(ph)
133     }else if (phase_com == 24){
134       original_phase_count<- 5
135       ph1<- max(yyy[c(T,F,F,F,F,F,F,F,F),3],yyy[c(F,T,F,F,F,F,F,F,F),3])
136       ph2<- max(yyy[c(F,T,F,F,F,F,F,F,F),3],yyy[c(F,F,T,F,F,F,F,F,F),3])
137       ph3<- max(yyy[c(F,F,F,T,F,F,F,F,F),3],yyy[c(F,F,F,T,F,F,F,F,F),3])
138       ph4<- max(yyy[c(F,F,F,F,T,F,F,F,F),3],yyy[c(F,F,F,F,T,F,F,F,F),3])
139       ph5<- max(yyy[c(F,F,F,F,F,T,F,F,F),3],yyy[c(F,F,F,F,F,T,F,F,F),3])
140       ph<- c(ph1,ph2,ph3,ph4,ph5)
141       vs_sum<- sum(ph)
142     }else if (phase_com == 31){
143       original_phase_count<- 6
144       ph1<- max(yyy[c(T,F,F,F,F,F,F,F,F,F),3],yyy[c(F,T,F,F,F,F,F,F,F,F),3])
145       ph2<- max(yyy[c(F,T,F,F,F,F,F,F,F,F),3],yyy[c(F,F,T,F,F,F,F,F,F,F),3])
146       ph3<- max(yyy[c(F,F,F,T,F,F,F,F,F,F),3],yyy[c(F,F,F,T,F,F,F,F,F,F),3])
147       ph4<- max(yyy[c(F,F,F,F,T,F,F,F,F,F),3],yyy[c(F,F,F,F,T,F,F,F,F,F),3])
148       ph5<- max(yyy[c(F,F,F,F,F,T,F,F,F,F),3],yyy[c(F,F,F,F,F,T,F,F,F,F),3])
149       ph6<- max(yyy[c(F,F,F,F,F,F,T,F,F,F),3],yyy[c(F,F,F,F,F,F,T,F,F,F),3])
150       ph<- c(ph1,ph2,ph3,ph4,ph5,ph6)
151       vs_sum<- sum(ph)
152     }
153     print(vs_sum)
154     sum_vs_old <- append(sum_vs_old, vs_sum)
155
156   }
157   vs_per_phase<- as.data.frame(matrix(ph,col = original_phase_count, byrow = TRUE), stringsAsFactors = FALSE)
158   com_phasing<- cbind(data.frame(EWNS,sum_vs_old,vs_per_phase)
159   return(com_phasing)
160 }
  
```

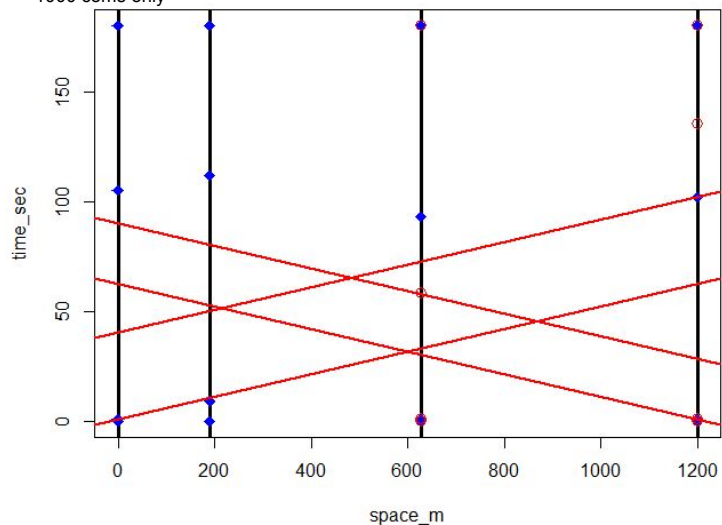


Process 1: 1-(1,2)>3>4>3> end

Process 2: 1-(1,2)>3>4>1-(1,2)>3>end

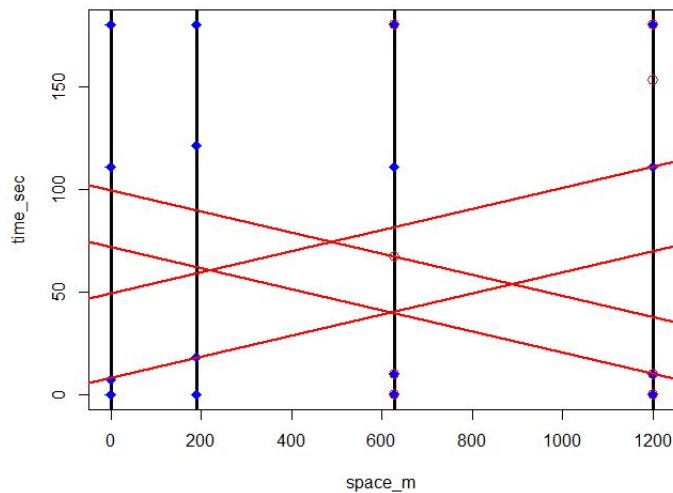


Before updating the green_split for 1000 coms only



```
> print(width_e)
[1] 39.28571
> print(width_w)
[1] 27.58286
> print(slag_down)
[1] 0.000000 1.771429 32.297143 61.714286
> print(slag_up)
[1] 64.71429 61.94286 20.41714 0.00000
> print(slag_down_w)
[1] 61.71429 43.94286 29.41714 0.00000
> print(slag_up_w)
[1] 14.70286 31.47429 0.00000 106.41714
```

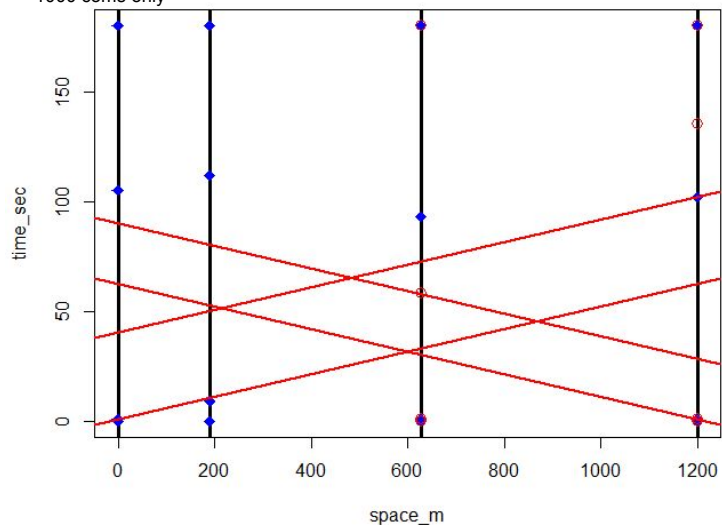
After updating the green_split



```
> print(width_e)
[1] 41.05714
> print(width_w)
[1] 27.58286
> print(slag_down)
[1] 1.228571 0.000000 30.525714 59.942857
> print(slag_up)
[1] 61.71429 61.94286 29.41714 0.00000
> print(slag_down_w)
[1] 6.471429e+01 4.394286e+01 2.941714e+01 7.105427e-15
> print(slag_up_w)
[1] 11.70286 31.47429 0.00000 115.41714
```

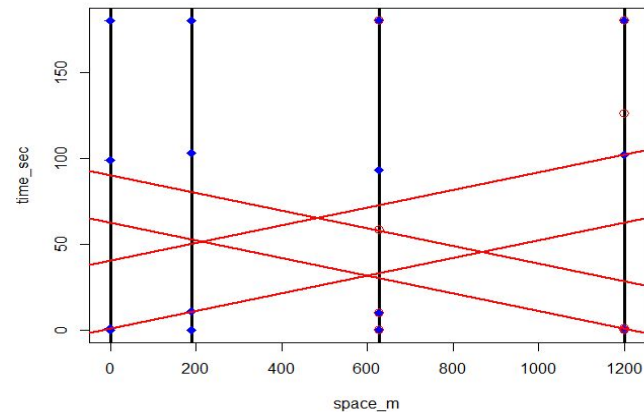
Updated

Before updating the green_split for 1000 coms only



```
> print(width_e)
[1] 39.28571
> print(width_w)
[1] 27.58286
> print(slag_down)
[1] 0.000000 1.771429 32.297143 61.714286
> print(slag_up)
[1] 64.71429 61.94286 20.41714 0.00000
> print(slag_down_w)
[1] 61.71429 43.94286 29.41714 0.00000
> print(slag_up_w)
[1] 14.70286 31.47429 0.00000 106.41714
```

After updating the green_split



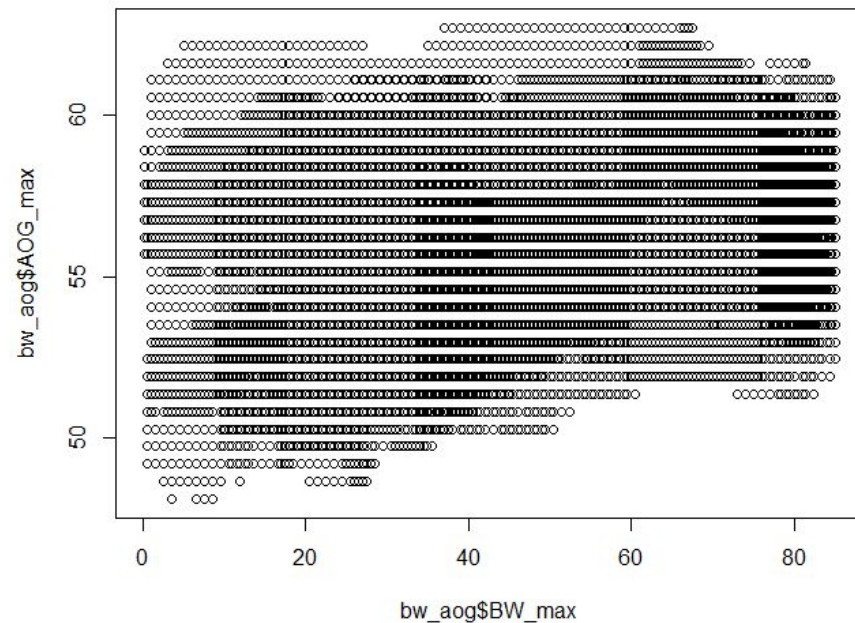
```
> print(width_e)
[1] 39.28571
> print(width_w)
[1] 27.58286
> print(slag_down)
[1] 0.00000 0.00000 23.29714 61.71429
> print(slag_up)
[1] 58.71429 52.94286 20.41714 0.00000
> print(slag_down_w)
[1] 61.71429 42.17143 20.41714 0.00000
> print(slag_up_w)
[1] 8.702857 22.474286 0.000000 97.417143
```

| | 교차로명 | 요일 | 시간 | phase1 | phase2 | phase3 | phase4 | phase5 |
|---|-----------|----|-------------|--------|--------|--------|--------|--------|
| 1 | 서대전나들목삼거리 | 주중 | 17:00~21:00 | 53 | 104 | 23 | NA | NA |
| 2 | 서일고교삼거리 | 주중 | 17:00~21:00 | 47 | 103 | 30 | NA | NA |
| 3 | 진잠네거리 | 주중 | 17:00~21:00 | 35 | 57 | 18 | 50 | 20 |
| 4 | 구봉중삼거리 | 주중 | 17:00~21:00 | 33 | 101 | 46 | NA | NA |

rearrange_total.R × phasing ×

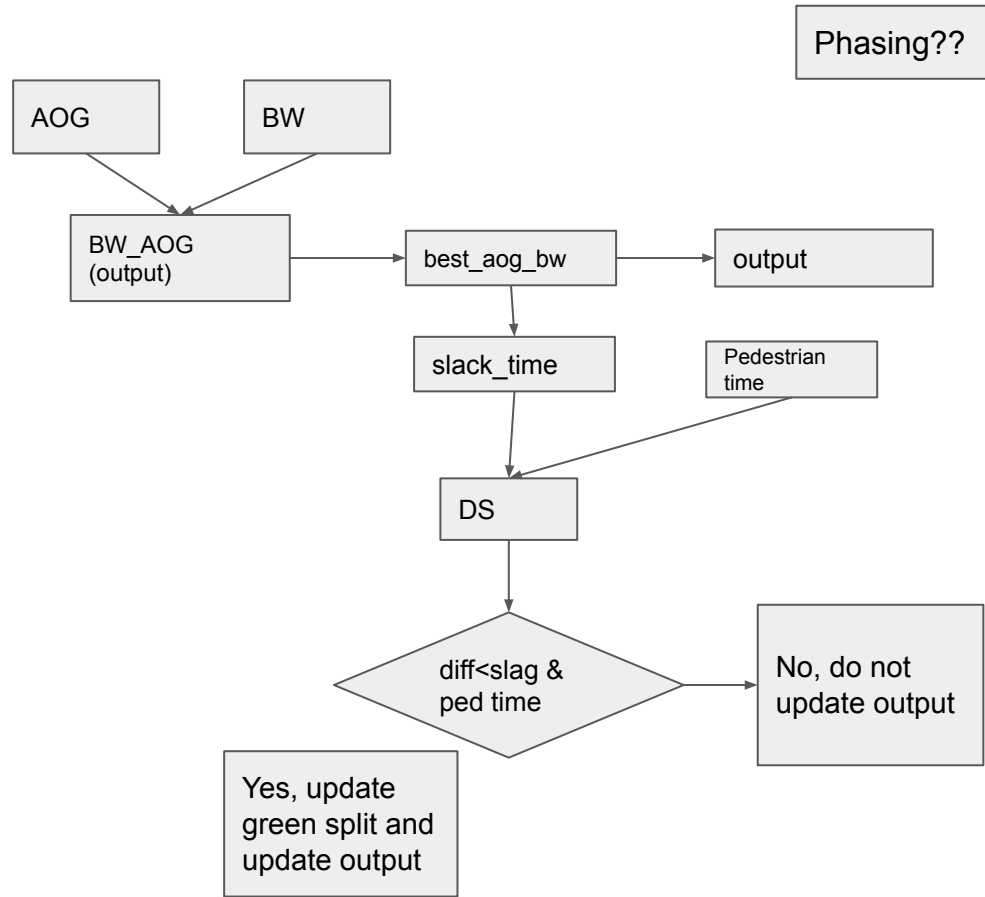
| | 교차로명 | 요일 | 시간 | phase1 | phase2 | phase3 | phase4 | phase5 |
|---|-----------|----|-------------|--------|--------|--------|--------|--------|
| 1 | 서대전나들목삼거리 | 주중 | 17:00~21:00 | 53 | 98 | 23 | NA | NA |
| 2 | 서일고교삼거리 | 주중 | 17:00~21:00 | 48 | 94 | 30 | NA | NA |
| 3 | 진잠네거리 | 주중 | 17:00~21:00 | 44 | 48 | 18 | 50 | 20 |
| 4 | 구봉중삼거리 | 주중 | 17:00~21:00 | 33 | 92 | 46 | NA | NA |

Final plot



```
> print(rec_com)
      com1 com2 com3 com4
169494    1    9   30    3
> |
```

| | | | | |
|---|------|------|------|--|
| | off1 | off2 | off3 | |
| 1 | 8 | 29 | 2 | |



BW_Input:

1. **Green time**
2. Distance between intersections
3. Phasing scheme
4. Speed for east and west bound

AOG_Input:

1. **Green time**
2. Phasing scheme
3. AOG arrivals on B.S during Cycle time

DS_Input:

1. Green time
2. Volumes
3. Lanes of road
4. Width of road
5. Phasing scheme
6. **Slack time**
7. Standard phasing number