Defragmenting algorithm

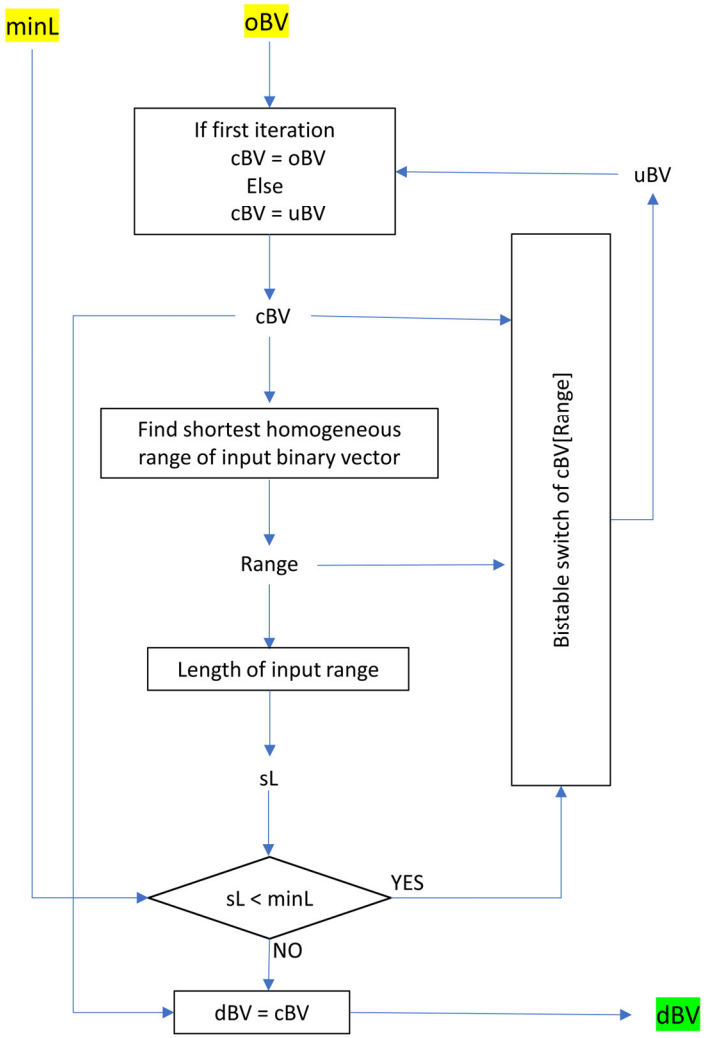
Parent article “Negative trend of regularity of locomotion in an endurance walking task: experimental data from healthy adult recreational athletes in an unsupervised 100-km march”, authored by Marco Rabuffetti, Ilaria Carpinella, Stefan Mendt, Giampiero Merati, Mathias Steinach and Martina Anna Maggioni

Given a binary vector (a data vector including binary values, such “0” and “1”, in any numerosity and order, including sequences of equal values) and given a unique parameter which is the minimal acceptable number of consecutive equal values, the defragmenting algorithm aims at modifying the original data vector by purging isolated values (i.e. values which are preceded and followed by the contrary value) or sequences of consecutive values which are preceded and followed by the contrary values.

The algorithm works iteratively by progressively identifying the shortest homogeneous sequence of values and substituting that sequence with the contrary values.

The iterations stop when no sequence shorter than the threshold value can be detected.

A flowchart showing the operations' sequence and feedback path of information, is presented here:



Defragmenting algorithm

Input Variables
oBV: original binary vector
minL: min length of homogeneus binary vector range

Internal Variables
cBV: current binary vector
uBV: updated binary vector
Range: pointer of adjacent vector cells
sL: shortest length of homogeneus adjacent vector cells

Output Variable
dBV: defragmented binary vector

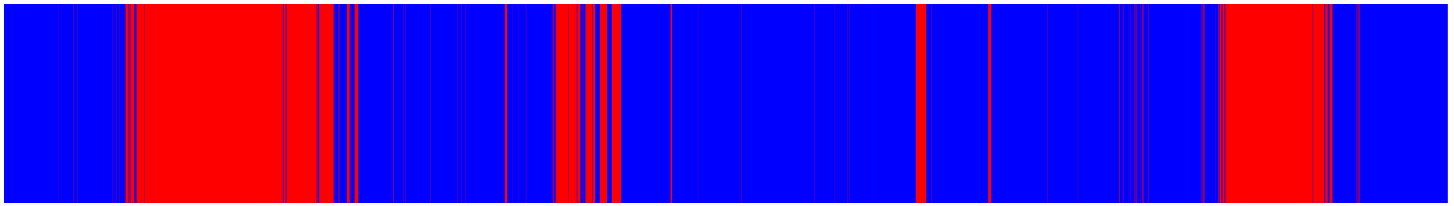
An example of data and the progressive modification introduce by the defragmenting algorithm until the final goal is reached follows

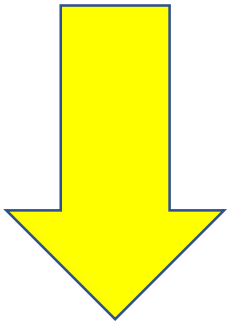
minL=5

oBV	1	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1
cBV	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1
cBV	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1
cBV	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	0	0	1
cBV	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1
dBV	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1

The original binary vector (oBV) includes, during its progressive modification, spurious occurrences of isolated single values, one doblet and one triplet. The final result (dBV) is a binary vector, closely related to the original binary vector, which does not include any homogeneous sequence shorter than the minimum value (minL). The constraint of an example to be shown on a page does not allow to evidence a peculiar aspect of the algorithm: a value which has been changed during one algorithm step, may be further changed in a subsequent algorithm step. The iterative nature of the algorithm enables such feature.

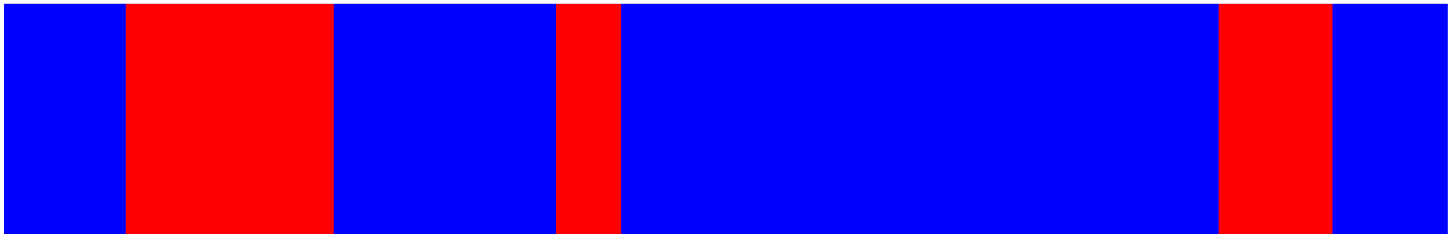
An example of true data coming from experimental data considered in the parent article is shown in the following figure. It is a blow up of a data section characterized macroscopically by the end of a previous walking bout, a resting period at one checkpoint, the following entire walking stage, the next resting period at a checkpoint, the start of the following walking stage. Red marks the resting epochs, blue marks the walking epochs. In the data set, epochs last one minute. It is evident how spurious epochs characterized by walking were present during the resting, and, vice versa, spurious resting (or no periodic behaviour) are present inside the walking stages. The final result shows up to be defragmented and the phases features (walking or resting) are more clearly defined.





The defragmenting algorithm iteratively and progressively eliminates the shortest epochs which are not homogeneous with preceding and following longest epochs sequences.

As a final result, three resting phases are identified, the first and the third red stripes are corresponding to resting at checkpoints, while the intermediate one was a pretty long resting along the path of a single walking stage



Finally, to evidence how the defragmenting algorithm, while simplifying the performance profile as composed by alternated walking and resting phases, does not affect the computation of the regularity and cadence outcome indices, an alternatiuve Table 2 from the parent article is proposed. The Table presented in the parent article was obtained by setting the algorithm parameter, the time threshold, to 300 seconds. The Table shown in the following was obtained by setting the algorithm parameter to 30 seconds. The indices related to the number and features of walking/resting bouts are obviously changed, while the indices related to regularity (RI0, RIend, mRI) and cadence (CI0, CIend,mCI) don't.

Id	Total Duration	Walking Duration	Resting Duration	Walking Bouts	Longest Bout	Longest Rest	RI_0	m_{RI}	RI_{end}	CI_0	m_{CI}	CI_{end}
1	21.86	18.53	3.33	32	3.87	0.77	0.950	-0.0001	0.948	107.8	0.14	111.0
4	22.13	19.70	2.43	18	3.58	0.71	0.954	-0.0037	0.871	108.8	-0.57	96.3
3	22.20	18.59	3.60	30	2.64	0.51	0.953	-0.0001	0.952	119.1	-0.14	116.0
6	23.03	20.28	2.75	27	2.98	0.50	0.970	-0.0012	0.943	115.3	-0.28	109.0
2	23.44	18.98	4.46	39	2.21	0.78	0.949	-0.0021	0.901	115.2	-0.20	110.5
7	25.22	20.43	4.78	42	3.89	0.91	0.938	-0.0023	0.879	114.1	-0.11	111.3
8	25.48	20.63	4.85	41	3.89	0.90	0.948	-0.0016	0.908	110.9	-0.13	107.4
9	25.78	20.78	5.00	48	1.71	0.88	0.953	-0.0011	0.925	113.3	-0.30	105.6
10	14.78	10.93	3.85	20	3.96	0.78	0.970	-0.0035	0.918	110.0	-0.37	104.5
5	12.23	10.69	1.54	14	2.14	0.34	0.965	-0.0046	0.909	117.5	-0.52	111.2