Enhancing semantic belief function to handle decision conflicts in SoS using k-means clustering

Eman K. Elsayed [Equal first author, 1, 2] , Ahmed Sharaf Eldin Ahmed [3, 4] , Hebatullah Rashed Younes [Corresp. Equal first author, 3]

1 Mathematical and Computer Science Department, Faculty of Science, Al-Azhar University, (Girls Branch), Cairo, Egypt

2 Computer Science Department, Faculty of Information Technology, Misr University for Science and Technology MUST, 6th of October, Egypt

3 Information Technology Department, Faculty of Information Technology and Computer Science, Sinai University, Arish, Egypt

4 Information System Department, Faculty of Computers and Artificial Intelligence, Helwan University, Cairo, Egypt

Corresponding Author: Hebatullah Rashed Younes
Email address: heba.rashed@su.edu.eg

Background: The endeavouring to offer complex special functions from individual systems gave rise to what is known as the System of Systems (SoS). SoS co-integrating systems together while allowing for absorbing more systems in the future. SoS as an integrated system simplifies operations, reduces costs, and ensures efficiency. However, conflict may result while co-integrating systems, violating the main benefits of SoS. This paper is concerned with enhancing the time required to detect and solve such conflicts.

Methods: We adopted the k-means clustering technique to enhance the detection and solving of conflict resulting while co-integrating new systems into an existing SoS. Instead of dealing with SoS as a single entity, we partition it into clusters. Each cluster contains nearby systems according to pre-specified criteria. We can consider each cluster a Sub SoS (S-SoS). By doing so, the conflict that may arise while co-integrating new systems can be detected and solved in a shorter time. We propose the Smart Semantic Belief Function Clustered System of Systems (SSBFCSoS) which is an enhancement of the Ontology Belief Function System of Systems (OBFSoS).

Results: The proposed method proved the ability to rapidly detect and resolve conflicts. It showed the ability to accommodate more systems as well, therefore, achieving the objectives of SoS. In order to test the applicability of the SSBFCSoS and compare its performance with other approaches, two datasets were employed. They are (Glest & StarCraft Brood War). With each dataset, 15 test cases were examined. We achieved, on average, 89% in solving the conflict compared to 77% for other approaches. Moreover, it showed an acceleration of up to proportionality over previous approaches for about 16% in solving conflicts as well. Besides, it reduced the frequency of the same conflicts by approximately 23% better than the other method, not only in the same cluster but even while combining different clusters.

1 # Enhancing Semantic Belief Function to Handle
2 # Decision Conflicts in SoS Using K-means Clustering

4 Eman K. Elsayed [1,2], Ahmed Sharaf Eldin Ahmed [3,4], Hebatullah Rashed Younes [3]
5 [1] Mathematical and Computer Science Department, Faculty of Science, Al-Azhar University,
6 (Girls Branch), Cairo, Egypt
7 [2] Computer Science Department, Faculty of Information Technology, Misr University for
8 Science and Technology MUST, 6th of October, Egypt
9 [3] Information Technology Department, Faculty of Information Technology and Computer
10 Science, Sinai University, Arish, Egypt
11 [4] Information System Department, Faculty of Computers and Artificial Intelligence, Helwan
12 University, Cairo, Egypt
13
14 Corresponding Author:
15 Hebatullah Rashed Younes [3]
16 Mansoura, Egypt
17 Email address: heba.rashed@su.edu.eg

19 ## Abstract

20 **Background:** The endeavouring to offer complex special functions from individual systems
21 gave rise to what is known as the System of Systems (SoS). SoS co-integrating systems together
22 while allowing for absorbing more systems in the future. SoS as an integrated system simplifies
23 operations, reduces costs, and ensures efficiency. However, conflict may result while co-
24 integrating systems, violating the main benefits of SoS. This paper is concerned with enhancing
25 the time required to detect and solve such conflicts.
26 **Methods**: We adopted the k-means clustering technique to enhance the detection and solving of
27 conflict resulting while co-integrating new systems into an existing SoS. Instead of dealing with
28 SoS as a single entity, we partition it into clusters. Each cluster contains nearby systems
29 according to pre-specified criteria. We can consider each cluster a Sub SoS (S-SoS). By doing
30 so, the conflict that may arise while co-integrating new systems can be detected and solved in a
31 shorter time. We propose the Smart Semantic Belief Function Clustered System of Systems
32 (SSBFCSoS) which is an enhancement of the Ontology Belief Function System of Systems
33 (OBFSoS).
34 **Results:** The proposed method proved the ability to rapidly detect and resolve conflicts. It
35 showed the ability to accommodate more systems as well, therefore, achieving the objectives of
36 SoS. In order to test the applicability of the SSBFCSoS and compare its performance with other
37 approaches, two datasets were employed. They are (Glest & StarCraft Brood War). With each
38 dataset, 15 test cases were examined. We achieved, on average, 89% in solving the conflict
39 compared to 77% for other approaches. Moreover, it showed an acceleration of up to
40 proportionality over previous approaches for about 16% in solving conflicts as well. Besides, it

41   reduced the frequency of the same conflicts by approximately 23% better than the other method,
42   not only in the same cluster but even while combining different clusters.
43   **Keywords** System of Systems (SoS); Component Systems (CS); Conflict; Clustering; k-means.

Introduction

46   Recently and according to systems, it has been proven that in order to create a powerful, co-
47   integrated, and multitasking system, an individual system will not be sufficient (Boehm and Lane
48   2007) (Robinson, Pawlowski et al. 2003) (Viana, Zisman et al. 2017). So, to achieve such a
49   system, some systems must be co-integrated together in System of Systems (SoS) (Spanoudakis
50   and Zisman 2001). SoS is a set of dedicated systems that have fetched their abilities to create a
51   new complicated system, providing more performance and services than individual systems
52   (Popper, Bankes et al. 2004). Organizations are continually facing challenges to co-integrate new
53   Component Systems (CS) and update existing systems over a while under threats, restricted
54   budget, and uncertainty (Agarwal, Dagli et al. 2016). Both complexity and uncertainty are
55   inherent features of infrastructure SoS, which can drive the operation of such systems away from
56   their intended purposes (Peculis and Shirvani 2017).
57   It is necessary for a successful system to accurately define the interests, objectives, and
58   requirements of its components, therefore, there are some limitations. SoS has been facing
59   different challenges due to the configuration and functions of the systems that make SoS up. One
60   of such challenges is managing goals to make an appropriate decision on SoS.
61   CS that participates in SoS arrangements might cause conflicting individual objectives among
62   themselves. As well as, the emergence of conflicting objectives between the entire SoS, it is CS
63   (Sage 2003).
64   Our objective is to have more CS within a single SoS, however, such systems must be
65   harmonious. A conflict is an issue that occurs between two or more systems in SoS, and it
66   represents a fundamental issue that has been extensively discussed for SoS. Such an issue might
67   be security, pattern, classifications, or decision. Resolving the conflict does not occur through
68   one stage but more. First, identify and detect the conflict, then diagnose to fix it appropriately
69   (Ramsbotham, Miall et al. 2011).
70   Multi-systems conflicts are a generalization of a single systems conflict problem for more than
71   one system, hence, in SoS, the more CS, the more conflicts. Several algorithms have been
72   proposed to deal with conflicts among CS in SoS, of which most deal with any CS in SoS as a
73   single system. In such cases, the aim is finding a conflict resolution for different systems,
74   guaranteeing that the given solution is optimal, and managing to handle a specified number of
75   CS.
76   This paper aims to find a better solution to the conflict problem. Better solutions are usually
77   applied when SoS CS are individuals. The mission is to attract and contain more SoS and get
78   better conflict resolution out of treating SoS CS as a joint system in clusters, and in turn, the
79   concept of clustering, and time reduction is important from a semantic point of view.

80 Clustering is a learning method that organises objects that are with similarities in one or more
81 features into clusters (Mokhtarpour and Stracener 2014), aiming to isolate objects with similar
82 traits and assign them into clusters. Consequently, eventually, each cluster should include
83 identical items which are different from those of other clusters.
84 Clustering algorithms are unsupervised pattern-learning algorithms (Wu, Xu et al. 2020) without
85 prior information, through creating smaller clusters with high intergroup dissimilarities and
86 intragroup similarities. A large number of these algorithms found. Hierarchical, Partitional, and
87 Bayesian are popular ones. This kind was purposefully chosen since SoS is a learned-system, and
88 unsupervised learning as a learning technique has permitted the model to work with its discovery
89 information; mainly deals with unclassified data.
90 The proposed approach aimed to adopt as many CS as possible into clusters, get such clusters to
91 form a significant SoS. Successful harmonisation of CS together in an SoS requires quick and
92 appropriate handling of conflicts.
93 There are some contributions to this paper as follows:
94 – Introduction of a new method to add more systems to SoS.
95 – Improving the handling techniques of the issues especially conflicts.
96 – Illustration of the potential of the proposed method to group more systems into clusters in
97   what resembles Sub SoSs (S-SoSs) within a large main SoS named General SoS (G-SoS).
98 – Demonstration of how the clustering method was more appropriate to cover objectives and
99   influence results, that is not in attracting more systems, but rather in reducing the number of
100   conflicts, which has, in turn, led to more time reduction enhanced the overall results.
101 This paper is organised as follows;
102 Part 1 covers a brief review of the conflicts in SoS, categorises all existing work into some main
103 categories (Decision, design, security and code), orders the different approaches for solving this
104 problem, then, in turn, displays some clustering techniques with related works. The main focus
105 was on the method, which has been developed for use.
106 Part 2 presents a way to attract more system and optimize conflict resolution in SoS, explains the
107 new algorithm (SSBFCSoS) that divided into two parts (levels). The single CS has initialised
108 with their default (working) task, which may contain conflicts, then use k-means clustering to
109 make them S-SoSs for the G-SoS.
110

Background and Related Works

112 The System of Systems (SoS) includes complex components and complex calculations (Gorod,
113 White et al. 2014), with challenges of which the most significant are design, development, and
114 decision. The detailed intricate design of SoS requires managing its CS to take advantage of it,
115 which in turn necessitates CS interoperability rather than compatibility with diverse assets
116 (Luzeaux 2014) (Lane and Boehm 2019), which does not usually fit the operational needs of the
117 large SoS. CS of SoS must be able to exchange data or information among themselves, either for
118 its purposes or for integration and fusion among them, and eventually the exchange process
119 between the architecture of different systems, that may cause design conflicts.

120   A survey in 2010 (Engineering 2010) explained the information and construction of SoS from an
121   engineering standpoint, combining various designs of CS that often cause conflicts. The methods
122   used often create conflicts as well, and clarify how data managed between CS and how to face
123   conflicts, through discovering another source of data in the event of a conflict. Such process
124   presented a new conflict if no other source was available, rather than a solution to the conflict,
125   contrary to what our research paper has provided, as in using BF to review source reliability,
126   depending upon the trusted source system.
127   Various security priorities of multiple CS in SoS can cause conflicts (Bodeau 1994), therefore,
128   the available security mechanisms may not meet specific SoS security requirements, leading to a
129   more complex task of integrating and architecture SoS (Maier 1998). If a particular system has to
130   interact securely with another CS system, its security architecture may have to deal with different
131   authentication mechanisms, presenting a security challenge in the SoS.
132   Some security issues of CS that make-up SoS have been discussed according to the
133   characteristics of SoS (Madan 2015), among which is integrating the different security structures
134   of the CS (Chiprianov, Gallon et al. 2014), which requires high characteristics of the SoS to
135   maintain the system, and guaranteeing the services despite some security problems, in terms of
136   networks and others.
137   The research (Ki-Aries, Faily et al. 2018) discussed an approach that coordinates the concepts of
138   SoS for analyzing security conflicts, depending on the independent collaboration of CS in SoS.
139   The research has explained that unknown information elements have frequently impeded the risk
140   assessment and decision-making regarding SoS. Contrary to our research, where the BF equation
141   has been used for this, to verify that – assumed unknown – information, which has enabled us to
142   make an appropriate decision for SoS, regardless of security conflicts either in this or other
143   research.
144   The conflict-resolution algorithm has been introduced for multi-agent systems related to
145   relationships of different types and values (Garanina, Sidorova et al. 2015) to obtain conflict-free
146   factors and solve ontology-based ambiguity in analyzing the natural language text. The algorithm
147   has relied on an agent weighting conflict resolution. Contrary to our research, which has relied
148   on the decision weight and both its accuracy and reliability, leading to more comfortable and
149   faster processing of complex systems since it depends on addressing the conflicting data rather
150   than the system or agent.
151   For systems containing databases, the time factor is essential, therefore as per the research
152   (Ahmed, Salem et al. 2015), more time may be needed to obtain data from the site, especially
153   from a distance. The process of clustering has helped to address this challenge. The concept was
154   reducing a large number of databases through clustering, as proposed by the research, for both
155   distribute data and systems, reducing the number of conflicts within SoS, and reducing the mass
156   conflict, according to the reliability estimate. It includes performance improvement and decision-
157   making while saving time.
158   Researchers (Wu, Xu et al. 2020) have proposed a clustering algorithm evaluation model to
159   assess conflicting performance, to reconcile the differences in the evaluation performance of

160 these algorithms. These conflicts have been shown during the process of integrating the
161 information during the decision-making process. Moreover, the proposed model can reconcile
162 conflicting evaluation performance to reach a collective agreement in the complex decision-
163 making environment.
164 Researchers have used the algorithm k-means to reduce the data dimensions (Idrees and Gomaa
165 2020), proposing a two pillars method. The first pillar is to define the consistency of traits more
166 precisely, as in the aspect of applying clustering techniques to remove less likely features.
167 Based on the kernel k-mean clustering algorithm, a proposed study (Muflikhah, Widodo et al.
168 2020) has shown that the algorithm could detect disease. Two types of datasets were extracted
169 from DNA lysis and grouped into two groups using the kernel k-mean algorithm, then the
170 classifier was applied to each group. When datasets aggregated, each group has significant
171 similarities in characteristics. The block dependent classification contains more support vectors
172 than the no-group category. Block quality also affects detection performance.
173

Clustering and its techniques

175 Clustering is a significant unsupervised learning problem, which draws references from datasets
176 consisting of input data without specific responses, and is used as a process to find meaningful
177 structures (Rokach and Maimon 2005).
178 There are no specific criteria for exemplary assembly, however, it depends on the appropriate
179 standards for users to meet their needs. For instance, in our method, we have worked on finding
180 representatives for Homogeneous Groups for better performance and adopted clustering to
181 reduce working all systems together. Contrary to our approach, the previous techniques have
182 detected conflicts on all CS of SoS, while we have used clustering to divide these CS into small
183 clusters, leading to not only more systems attraction, but better performance and conflict
184 resolution.
185 **Clustering methods are divided into three parts.**
186 **a. Hierarchical:** The algorithms of this technique depend on finding its clusters using pre-
187   created ones. It can be either agglomerative or divisive (Rokach and Maimon 2005).
188   Agglomerative (bottom-up) starts with each object as a separate cluster and then merges
189   them into larger clusters while Divisive (top-down) begins with the whole set, then divides it
190   into smaller clusters.
191 **b. Partitional:** The algorithms here define all clusters then use the Divisive algorithms
192   technique for grouping. It contains many techniques as Model-Based, Graph-Theoretic, and
193   Spectral. Besides, Centroid techniques to which the k-means algorithm belongs.
194   Clustering deals mostly with data segmentation within mega systems. Contrary to our
195   research paper, which depends on dividing the systems within SoS rather than the data, the k-
196   mean technique has been used for the clustering process. As a partitional algorithm, it starts
197   with a fixed number of clusters. After determining the number of clusters, the class of object
198   n is determined (Rokach and Maimon 2005), then assigned to the nearest cluster centre. The

199   centres of mass k are then re-estimated assuming that the above memberships are valid. This
200   process is repeated until n-objects membership does not change in the last iteration.
201   **c. Bayesian:** The algorithms of this technique generate posterior distribution to collect all data
202   partitions. It has two algorithms decision Based and Nonparametric.
203

Smart Sematic Belief Function Clustered System of Systems method (SSBFCSoS)

206   The ability of SoS to perform large tasks lies in coordinating its CS as a single system. The more
207   the SoS can assemble more systems and do this coordination, the better. Obtaining the optimum
208   decision from all systems while maintaining security is the goal of SoS. SoS is constructed from
209   many CS, which conflicts among them cause SoS to be inefficient. In 2019, researchers (Younes,
210   Ahmed et al.) have adopted a method to get the best Decision for SoS while avoiding conflicts
211   between CS. The algorithm used was ontology with the Belief Function equation that modified
212   to suit systems. Learning has limited SoS capacity to obtain the best results, since during
213   running, and when CS reaches a specified number, SoS gets more time to handle conflicts. As a
214   learned-system, it represents a challenge, so, an SSBFCSoS algorithm that depends on the
215   clustering idea was suggested.

The Handled Clustering Technique for SoS in SSBFCSoS

217   Clustering is the idea for SSBFCSoS to divide CS, not data into clusters. We have primarily
218   relied on the use of the k-means method, and a combination of this method and another
219   agglomerative method (bottom-up). According to the used k-means method; defining each
220   cluster according to its properties requires creating clusters that share common properties. We
221   adopted an idea that depends on the location of the CS systems, hence each cluster contains
222   nearby systems according to pre-specified criteria.
223   While in the case of partitional clustering k-means, the k partitions created containing n objects,
224   each partition represents a cluster created, where $k \leq n$, attempting to divide items into clusters
225   based on some evaluation criteria. A specific number of clusters k determined by a set of
226   centroids (initially assigning a centroid to each cluster) where $C = \{C_1,...,C_k\}$. The centroid
227   calculation for each cluster according to the mean M_k of all the objects N_k that belong to this
228   cluster as in equation 1.

$$M_k = \frac{1}{N_k}\sum_{q=1}^{N_k} X(q) \qquad (1)$$

230   The members define each centre. Moreover, if the properties are incompatible, re-merge them
231   again with any other appropriate cluster. Furthermore, this matter is repeated until each object
232   belongs to a suitable cluster. Moreover, the stability of the clusters and their centres occurs
233   without change. Each cluster is to be considered as Sub SoS (S-SoS) that contain systems as
234   objects.

235 Determining the appropriate S-SoS to belong to a particular system depends on how close the
236 traits are between systems of the same S-SoS. The primary is the close locations between
237 systems in each of S-SoS. The k-means algorithm measures the extent of similarity by
238 calculating distance. Both distance and similarity are related. Distance measure determines the
239 similarity between items and affects the shape of clusters, a smaller distance between two objects
240 results in more similarity.
241 Initialising S-SoSs is achieved by picking one point per the first S-SoS randomly as an initial
242 centroid point, then $k - 1$ for the other S-SoSs; each centroid point is as far away as possible
243 from the previous centroid ones.
244 Depending on the k-means method (Solanki and Pittalia 2016), the distance is calculated
245 between each system of the S-SoS to every centroid initialised. Based on the values found,
246 systems are assigned to the centroid with the lowest distance.
247 There are four types of distances to form clusters to measure as follows;
248 **Euclidean distance measure**
249 Assuming that there are two systems p and q, the Euclidean distance is a standard straight line
250 between them. The calculation is according to equation 2.

251
$$d = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \qquad (2)$$

252 **Squared Euclidean distance measure**
253 It is identical to the Euclidean distance measure but does not take the square root.

254
$$d = \sum_{i=1}^{n}(q_i - p_i)^2 \qquad (3)$$

255 **Manhattan distance measure**
256 The absolute difference between the two systems sum computed, coordinates are shown in
257 equation 4. It is the distance between two systems measured along axes at right angles.

258
$$d = \sum_{i=1}^{n}|q_x - p_x| + |q_y - p_y| \qquad (4)$$

259 **Cosine distance measure**
260 The angle cosine between the two systems vectors is determined in the space of dimension n, as
261 in equation 5, considering that p_i and q_i are components of vector p and q.

262
$$d = \frac{\sum_{i=0}^{n-1} q_i - p_i}{\sum_{i=0}^{n-1} q_i^2 \times \sum_{i=0}^{n-1} p_i^2} \qquad (5)$$

263 To form clusters (S-SoSs), first, random points as S-SoS called centroids are assigned. Second,
264 assign each system to the closest S-SoS centroid by implementing a Euclidean distance (the

off

265   system distance to centroid). Third, identify the new centroids by taking the average of the
266   assigned systems points, then continue to repeat the second and third steps until convergence is
267   achieved.
268   It is the first part of the SSBFCSoS algorithm. Conflicts are detected and addressed within each
269   subsystem in the OBFSoS algorithm (Younes, Ahmed, et al.), then comes the second part of the
270   clustering method; aggregating all S-SoSs into G-SoS, in stages that mainly depend on the
271   number of S-SoSs.
272   The last part of the clustering method in SSBFCSoS belongs to the agglomerative hierarchical
273   algorithm, through the bottom-up process. This algorithm is based on finding the distance from
274   the closest pair of S-SoS via the distance between the two centroids $d(S\_SoS_{AB})$. Assuming that
275   the two closest clusters are S-SoS (A) and S-SoS (B), the newly formed cluster would be S-SoS
276   (AB). Reset centroid for the new cluster, and update the distance matrix for these two clusters,
277   giving only a distance between the new S-SoS (AB) and the other remaining clusters, then
278   calculate the average distance between two clusters as the average distance between all systems
279   as in equation 6, where d_{ik} represents the distance (similarity). $N(S_{SoSAB})$ and $d(S\_SoS_C)$ are
280   numbers of systems in clusters A, B, and C.

281
$$d((S\_SoS_{AB}))_C = \frac{\sum_i \sum_k d_{ik}}{N_{(S\_SoS_{AB})} N_{S\_SoS_C}} \qquad (6)$$

282   Finally, detect conflicts inside the new cluster and address them using OBFSoS, then iterate
283   these steps until all clusters merged into a single one called General SoS (G-SoS) so that all
284   systems in G-SoS after the algorithm are terminated.

285   **Proposed Method**
286   The proposed method SSBFCSoS considers an enhancement of the OBFSoS method (Younes,
287   Ahmed et al.). Both methods have adopted the concept of better decision making for SoS by
288   addressing conflicts between CS. The proposed SSBFCSoS method consists of three phases,
289   summarized in Fig.1.
290   **The first phase** deals with k-means, one of the clustering techniques for dividing CS into
291   groups. Such groups use Java Format, Packages in Glest Game and StarCraft Brood War
292   packages. The requirements are; processor Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz   2.40
293   GHz, RAM 8.00 GB and platform is 64-bit Windows 10 operating system, x64-based processor.
294   **The second phase** presents the reversal of Glest and StarCraft Brood War code generating into
295   OWL Ontologies of all systems in each cluster, together with merging the OWL ontologies for
296   conflict detection and decision making for each S-SoS. Plus, StarCraft Brood War is converting.
297   **The third phase** conflict detection of conflicts while merging the S-SoSs using another type of
298   clustering technique; getting G-SoS better decision.
299   Coordinating systems with common characteristics using Ontology is easier (Gutierrez 2018),
300   where avoiding conflicts resulting from similar systems components. The Belief Function (BF)
301   equation assumes in the OBFSoS algorithm, which used for the SSBFCSoS method as well. BF

302   primary use is enabling the measurement of conflict weight, and the probability of decision
303   making.
304   The algorithm is in three parts, two of which are for the clustering processes and the third is for
305   conflict resolution within each cluster summarized in Algorithm 1, which presents the pseudo-
306   code for SSBFCSoS, depending on using two clustering methods; the (k-means) method and the
307   (bottom-up) agglomerative method.
308   **The first phase** specifies the maximum number of desired systems (s_i) in each cluster and the
309   number of desired clusters (C_l), then, after determining the number of clusters to be divided for
310   the systems to enter, centroids are were chosen randomly, including the rest of the systems in the
311   clusters, hence, calculating the distance (d_{s_i}) for connecting each system in the cluster with the
312   closest centroids.
313   Calculating the distance according to the location of each system is performed by several
314   methods. Euclidean distance measurement is the adopted method (equation 2 & 3), since it was
315   more suitable for the homogeneous cases applied, because of the distance calculation based on
316   the raw inputs.
317   The primary advantage of the Euclidean method is that the distance between any two systems is
318   not affected by adding new systems to the analysis, which may be outliers, however, distances
319   can be significantly affected by differences in scale between the dimensions over which
320   distances calculated. For example, if a dimension denotes the length measured in centimetres, it
321   is preferred not to convert it to another scale, as the resulting Euclidean can be significantly
322   affected. Euclidean distance gives the best result (Singh, Yadav et al. 2013).

323
$$J = \sum_{l=1}^{r} \sum_{i=1}^{n} \left\| s_i^{(l)} - C_l \right\|^2 \qquad (7)$$

324   Equation 7 calculates the distance for systems, considering r number of clusters, n number of
325   systems s_i case i *and* C_l the centroid for cluster l. J is an objective function where, $\left\| s_i^{(j)} - C_l \right\|^2$
326   represent the distance function.
327   After determining the inputs, all systems are considered part of a cluster (S-SoS), and the
328   selection of the cluster depends on the centroid focal point closest to it. One system in two
329   clusters is not allowed at the same time.
330   After organizing the clusters, each cluster will be addressed as a separate system, so that each
331   cluster will resolve conflicts between the CS, obtaining the best decision and observing all the
332   regulations and tasks related to it. All this is achieved in parallel to the clusters S-SoSs.
333   **The second phase** is the OBFSoS method role, which primarily depends on addressing conflicts
334   and obtaining the best decision. In this method, weight is relied upon because the system relies
335   on more data for decision-making, this means that it examines more data to make the right
336   decision. The requirement is to obtain a system decision with the most considerable value for
337   weight, suppose the weight values are equal in more than one system. In this case, the solution is
338   either merging the decisions or leaving the decision to the user.

339   The output variable of the OBFSoS method is X_{ij} for optimizing decision making for SoS, the
340   weight variable w_{ij}. X_{ij} represents each cluster decision. w_{ij} represents the weight of this
341   decision. Each cluster as a part of General SoS (G-SoS) needs the output maintained while
342   prioritising conflict issues according to SoS requirements, putting into consideration that
343   decision-making precedes delegation decisions.
344   **The third phase** includes obtaining the information required from each cluster X_{ij} and w_{ij},
345   followed by entering into the final stage of algorithm 1, to obtain the best Decision regarding G-
346   SoS. This is the stage of comparing the obtained results from the OBFSoS algorithm, comparing
347   the effects and weights of all decisions to enable the G-SoS to obtain the optimal decision.
348   Another kind of cluster approach has been adopted to finally integrate the S-SoSs into a single
349   SoS (G-SoS). At the end of the algorithm SSBFCSoS, the two closest clusters (S-SoS) are
350   identified by calculating min distance $d(i,j)$, then merging them with the OBFSoS algorithm as
351   in MS_{SoS}. Then, although the weights of each S-SoS are preserved, conflicts were detected in the
352   new S-SoSs that merged; a sequence repeated until a G-SoS idealized decision is reached.
353   There is only one particular case. where there are two groups of equal weights; the two clusters
354   decisions are combined allowing the human element to intervene and reach the right decision.
355

Case study 1 (Glest)

357   SSBFCSoS, as an algorithm represented in this paper, applies to many systems states. One of
358   these instances and the experimental setup used for our empirical analysis is described. This case
359   has been selected and set up to match previous work (Younes, Ahmed, et al.). Ontology is useful
360   for systems with similar components. Such systems can be called homogeneous systems. One of
361   these homogeneous systems is gaming. The system does not just mean having software, but
362   rather hardware and software, in addition to the human component and the communication
363   between all of these components. This online connection can be in the case of online games,
364   provided that every player with all these components is considered a system.
365   The choice was an action-based RTS (Real-Time Strategy) game called Glest (Dimitriadis 2009).
366   Several actions in the game represent data flows, and each group of actions belongs to a
367   category. It is impossible to deal with layers of the same name without conflict. The similarity of
368   names between categories does not necessarily indicate the similarity of the data.
369   Such similarity and the resulting conflict caused by it have to be avoided, which is the ultimate
370   role of Ontology, depending on the semantic comparison, which differentiates between the
371   classes meanings of the same name and content with a different owner. It is also necessary for
372   SoS to determine the most appropriate decisions. It is for SoS to perform its duties to the fullest.
373   The systems deal with more than one stage, of which one is the resulting resolution for each
374   cluster S-SoS. The other step is to obtain the final decision resulting from G-SoS. The goal is to
375   determine the winner as a decision, however, the first stage of its decision is the trade-off
376   between winning and action, as in every cluster that plays as an S-SoS has two directions.
377   Suppose each entry system for this S-SoS plays as a separate system, the S-SoS waits for the
378   decision of which systems to be the winner.

379 Moreover, more than one system within this group played as multiple, here the action
380 differentiated as a decision. Any action is the right one, and the winning decisions are compared
381 to others inside the cluster. The better the number of points paid, the better the decision. The
382 points here represent the data flow; consequently, the validity of the data.
383 The faster the SoS, the better. So the individual system is handled as an intelligent agent, to save
384 time because intelligence allows the system to avoid its mistakes, repeating them, and repeated
385 conflicts.
386

Case study 2 (StarCraft Brood War)

388 The StarCraft Brood War (Dor 2014) RTS game applied the same rules as Glest. The main focus
389 of this paper will be the final calculations of the conflict and resolution. The conflict of the
390 Brood War represents the differences between players of each cluster of G-SoS, to evaluate the
391 solution selection mechanism for it depending on algorithm 1.
392

Results and Discussion

Glest

395 Both SSBFCSoS and OBFSoS have been implemented. For SSBFCSoS, the systems were
396 divided into clusters. SSBFCSoS is very effective. The two algorithms are based on ontology and
397 BF use. However, SSBFCSoS with this clustering still outperforms OBFSoS in many scenarios.
398 Each attempt begins with 3 clusters, and each cluster includes 15 to 30 systems ranges. The
399 formation of clusters depends on the location of the systems, according to system IP with a time
400 limit of 120 seconds for both algorithms sets, which is for the OBFSoS algorithm to define
401 conflicts between systems. For SSBFCSoS, the time for each cluster is to identify the systems
402 involved to get conflicts. If an algorithm lacks the ability to resolve an issue within the time
403 limit, it stops, and the failure returned. The goal here is to study the behaviour of the two
404 algorithms for a specific number of systems.
405 Table 1 shows the number of conflicts generated for SSBFCSoS compared to OBFSoS averaged
406 70 systems. For ten attempts, in the case of SSBFCSoS, the number of conflicts for the input
407 systems is recorded, with a success rate more than the OBFSoS algorithm.
408 Fig.2 shows the conflict rate for SSBFCSoS compared to OBFSoS, which appears for 60 seconds
409 by the algorithms for several systems. In these minor problems, SSBFCSoS is inferior in conflict
410 rate to OBFSoS, having a slight advantage.
411 Different algorithms give an instance of a problem with a certain number of s_i systems. The
412 runtime also gives to two algorithms. Each algorithm is given 300 seconds to resolve the
413 conflicts it encountered. SSBFCSoS divides these systems into S-SoS (clusters) and implements
414 the algorithm associated with each sub-problem separately. Only final results for the SSBFCSoS
415 and OBFSoS algorithms are presented.
416 There is a sequence in the comparison values between SSBFCSoS and OBFSoS in resolving
417 conflicts regarding decision, design, security, and code. For the first method SSBFCSoS with

418   each cluster value before and after resolving conflicts for all Tables 2, 3, 4, and 5, in contrast to
419   the other method OBFSoS in which the conflicts and the resolution for the whole SoS are
420   without clustering.
421   The decision is to determine which system got the best points to win the game. That is with
422   regards to systems such as an RTS game as the Glest, as a result of making the right decisions.
423   As part of the algorithm, BF is used for the validity of action sources for each system (player),
424   and using a fewer number of right actions with the same result giving the player the right to win
425   the opponent, even if their points are equal, thus, the decision to win.
426   Table 2 shows the rate of conflicts in the decisions and their solutions between SSBFCSoS and
427   OBFSoS. The number of attempts denotes the simulation attempts column. For SSBFCSoS, the
428   number of desired clusters is indicated by C_l. For the given number of s_i systems (70), the best-
429   performing algorithm result is given in bold as in SSBFCSoS.
430   In Table 2, the values that resulted from applying clustering through SSBFCSoS has shown
431   better results. Attempt 5, for instance, shows the decrease of the conflicts percentage to 0.206 for
432   all the clusters in SSBFCSoS compared to 0.346 for the other method. The results have
433   progressed to reach 0.067 for SSBFCSoS compared to 0.357 for OBFSoS for the last attempt.
434   Fig.3 illustrates the rate of remaining conflicts in decisions while resolving the SSBFCSoS and
435   OBFSoS algorithms. It illustrates how SSBFCSoS encounters more conflicts on time, and is,
436   therefore, more suitable for SoS than OBFSoS.
437   Compared to OBFSoS, the SSBFCSoS algorithm curve shows that the rate of conflicts decreases
438   sequentially, as shown in Fig.3, due to the clustering of conflict handling. The system is an
439   intelligent, learned system that avoids the same conflicts encountered in earlier attempts. It
440   appears, as shown in Table 1, that applying the algorithm SSBFCSoS, the total number of
441   conflicts decreases as the number of attempts increases, so the relationship here is considered
442   inverse.
443   The system design is the general structure of the system. The parts that this system consists of
444   include units, interfaces, and data. As a system, the game also has a pattern or design.
445   Furthermore, from it, the incompatibility between the structure of two systems (two games) or
446   more, is what calls for design conflicts. Although systems are homogenous, different versions of
447   the game can cause such design conflicts. The same thing may cause code conflicts, as well.
448   The goal is to make the right decision to determine the winning player and also collect points. In
449   case there are design conflicts between two or more systems, the right decision cannot be made,
450   because of the incompatibility between the design of the two systems. According to the rules, to
451   determine the best work, the environment should be equal, since incompatible environments lead
452   to design conflicts, depending on the different versions of the game. If Algorithm is used,
453   SSBFCSoS determines the compatibility. BF also returns to the game source to verify this.
454   Table 3 displays the rate of conflicts in design with its solutions between SSBFCSoS and
455   OBFSoS. For example, in the last attempt, conflicts in the design percentage reached a ratio of
456   0.067 in SSBFCSoS compared to 0.243 in OBFSoS, which is considered a good result, as shown
457   in Table 3 that almost all the conflicts in two clusters were dealt with against one conflict in the

458   last cluster concerning method SSBFCSoS. While OBFSoS needed a longer time to reach the
459   same results. The clustering process dealt with conflicts in parallel, thus it consumed less time
460   than without it.
461   Fig.4 illustrates the rate of remaining conflicts in design after they encountered, during the
462   process of resolving the SSBFCSoS and OBFSoS algorithms. It describes how SSBFCSoS faces
463   more conflicts than OBFSoS in a limited time.
464   In Table 4, the results of the comparison of SSBFCSoS and OBFSoS shown in the code
465   conflicts. In addition to clarifying the clusters, each cluster is presented separately before and
466   after applying the SSBFCSoS algorithm. SSBFCSoS was able to face conflicts almost double the
467   other method. Whereas for SSBFCSoS, the conflict ratio has decreased to about 0.067 compared
468   to 0.114 for OBFSoS. Either way, it consumed about the same time and could be less.
469   Fig.5 shows the sequence of results for both algorithms to resolve conflicts in code. The higher
470   the number of attempts, the better the outcome for each algorithm, however, algorithm
471   SSBFCSoS still produces significantly better results as in the figure.
472   In regards to priority for each system, it may be a security priority. Some systems place security
473   at the top of their interests, according to their goals. In this paper, the systems are gaming
474   systems. Moreover, the game has an online version, therefore, security becomes essential. The
475   importance of security lies in its influence on the decision of our SoS. So, it is considered
476   working for security if it affects the decision because getting a better decision is the first and
477   primary goal.
478   The score has to be confirmed to determine who won, which is why, in this case, not all players
479   can enter the game until they pass the registration rules. That is to achieve the security
480   requirement. For that, Table 5 records the evolution in the number of conflicts in security. In
481   Table 5, SSBFCSoS achieved good results in facing security conflicts as much as possible. The
482   conflicts ratio was at attempt 7 was about 0.080 and 0.126 for OBFSoS for the same attempt.
483   Fig.6 shows the rate of development of security conflicts after applying methods, showing
484   satisfactory results by the SSBFCSoS in comparison to OBFSoS.
485   Table 6 shows the conflict rate resolved by the algorithms, and the time takes for each algorithm
486   to obtain these solutions. We have reached a better SSBFCSoS algorithm to get good results,
487   which consumes a shorter time. As shown in Table 6, the first attempt, the resolving conflicts
488   percentage for the SSBFCSoS has reached 0.296 within approximately 281 seconds, while
489   OBFSoS achieved 0.076 within 300 seconds. As for the last attempt, the rate of resolving
490   conflicts has reached 0.733 for the SSBFCSoS within 113 seconds, while OBFSoS has achieved
491   about 0.171 within 0.276 seconds.
492   These rates show in Fig.7, which reflects the rate of the conflicts resolved.
493   Fig.8 shows a comparison of the time taken for each algorithm to obtain solutions, proving that
494   the SSBFCSoS algorithm is the best in terms of time consumption.

StarCraft Brood War

496   To evaluate the StarCraft Brood War, the conflict resolution rates in the cases under
497   consideration generated by SSBFCSoS and OBFSoS are calculated and listed in Table 7. For

498 decision conflicts appearance as in attempt 9, the SSBFCSoS has reached 0.250 compared to
499 0.394 by OBFSoS. For design conflicts appearance as in attempt 8, the SSBFCSoS has reached
500 0.250 compared to 0.287 by OBFSoS. As for security conflicts in the same attempt, SSBFCSoS
501 has reached 0.167 compared to 0.204 by OBFSoS.
502 Table 8, shows that SSBFCSoS has achieved superiority over OBFSoS in terms of conflict
503 resolution in SoS.
504 Table 8 presents the number of conflicts generated for SSBFCSoS compared to OBFSoS with an
505 average of 100 systems for G-SoS. SSBFCSoS divides these systems into 3 clusters (S-SoSs),
506 with the number of systems ranging from 25 to 36 per cluster. Each algorithm is allowed 400
507 seconds to resolve the conflicts encountered.
508 Table 8 shows only the final result for the three merged clusters in decision, design, code and
509 security conflicts. At attempt 8, the percentage of conflicts after applying SSBFCSoS has
510 reached 0.208 compared to 0.389 for OBFSoS in decision conflicts. In design, conflicts have
511 reached 0.185 for SSBFCSoS compared to 0.242 for OBFSoS in attempt 7. In attempt 9, the
512 SSBFCSoS conflict ratio has reached 0.100 compared to 0.154 for OBFSoS in terms of security,
513 and the code conflicts ratio has reached 0.100 for SSBFCSoS compared to 0.163 for OBFSoS.
514 Fig.9 shows the improvement in the conflicts rate in decisions for SoS achieved by SSBFCSoS
515 in comparison to that achieved by OBFSoS.
516 Fig.10 shows that the conflicts in the design reduced by using SSBFCSoS better than OBFSoS.
517 Fig.11 with almost the same conflict rate for SSBFCSoS and OBFSoS. However, SSBFCSoS
518 can resolve more conflicts than OBFSoS.
519 The curve in Fig.12 illustrates the code conflicts of the two methods.
520 Table 9 shows the rate of addressing conflicts of the SoS in the ratio of both SSBFCSoS and
521 OBFSoS along with the time taken. In attempt 7, the percentage of solutions achieved by
522 SSBFCSoS has reached 0.333 within 127 seconds compared to a ratio of 0.70 within 317
523 seconds for OBFSoS.
524 Fig.13 shows the extent to which SSBFCSoS was better than OBFSoS in terms of addressing
525 conflicts, and its rate has reached 67% in attempt 10 compared to 17% for OBFSoS.
526 Fig.14 Compares the time spent for each method, showing the advantage of SSBFCSoS in saving
527 time at a rate of more than three times the time consumed by OBFSoS.
528 Each of the two systems revealed the number of conflicts. The results indicated the progress of
529 SSBFCSoS, which has reached this success thanks to the division of inputs from the systems to
530 fall under the name clusters S-SoS. It also shows an apparent effect on the proper use of time.
531 Focusing on time is essential in exceptional cases, such as the systems to which it was applied.

Comparing the two methods SSBFCSoS and OBFSoS

533 Experience clearly shows that there is no established law for making a better Decision for SoS.
534 However, the performance of each of the applied algorithms depends on the features of the
535 conflicts.

536 At the beginning of the application, a certain number - three - of clusters are identified, with each
537 cluster containing 20 to 35 systems, ultimately reaching about 70 or 100 systems. Time is set to
538 approximately 120 seconds to identify conflicts, and about 300 seconds to address those
539 conflicts. These features can be of conflict type and number of systems that have conflict at the
540 same time, in addition to the number of conflicts encountered during the solution process.
541 Each algorithm has its method depending on the case. Nevertheless, through research, a method
542 of evaluating the performance of each algorithm has been proposed. So, we present the following
543 general trends that we observed:

544 ▪ The algorithm SSBFCSoS outperforms the other algorithm in most cases with an
545 unlimited number of systems.
546 ▪ The longer the time is taken by algorithm OBFSoS, the better-achieved results, but to a
547 limited number of systems only.
548 ▪ As for the clusters in algorithm SSBFCSoS, the system must not be part of more than one
549 cluster simultaneously.
550 ▪ The game running online, allows the system to exit and enter more than once. It could
551 create a repeated conflict, especially in security.
552 ▪ Although the algorithm system SSBFCSoS is a learned system. However, it can seriously
553 increase in some attempts over others in conflicts for the previously mentioned reasons.
554 ▪ Continuous attempts with algorithm SSBFCSoS reduce the number of conflicts and the
555 time taken to resolve conflicts.
556 ▪ There is a constant conflict despite repeated attempts, which requires user intervention to
557 get the appropriate and better decision. Therefore, a significant percentage of about 20%
558 of the unresolved conflicts was required for user intervention.

559 Fig.15 shows the number of systems and clusters and the conflict resolution rate. The number of
560 clusters was set proportional to the number of systems, creating a direct relationship; the more
561 systems and clusters, the more satisfactory the results. Should we let the system limit the number
562 of clusters? If we leave it open, it is possible to create semi-empty clusters containing only two
563 or three systems. So, we tried to define the systems. During implementation, we found that if the
564 average number of systems within each cluster was 10% of the total number of systems, the
565 results were very appropriate, and the time was right. Also, if the average systems for each
566 cluster are 25-30, the results are perfect.

567 Also, we found a second situation during implementation where two or more systems (players)
568 play with the multiplayer system, as in the two systems produce one combined decision. First,
569 we need to compare procedures by OBFSoS algorithm, then compare their decisions. We have
570 two parts to compare the procedures for obtaining the highest points first and determining the
571 best procedure for obtaining higher points for their group, then compare the decisions to get an
572 ideal decision. And then they entered as part of the S-SoS and so on.

573 **Comparing clustering methods k-means and agglomerative (bottom-up)**

574 We made a statistic that compared two clustering algorithms, namely k-means and agglomerative
575 (bottom-up). The two algorithms were compared in the second case study; StarCraft Brood War.
576 We made this statistic on different numbers of systems, as shown in Table 10. We were
577 beginning with fewer systems, ending with more. We recorded the values after seven attempts.
578 Concerning the algorithm k-means, we specified the number of clusters as 5 clusters. Contrary to
579 k-means, the number of clusters in agglomerative (bottom-up) is not determined, but each system
580 is considered as a separate cluster until several clusters formed. The same distance measurement
581 is used for the k-means.
582 In Table 10 for case 7 with 110 systems as inputs, k-means achieved a conflict resolution rate of
583 about 90% compared to about 69% for the agglomerative (bottom-up). These results took about
584 279 seconds for the k-means compared to 790 seconds for the other one.
585 Fig.16 clarifies the comparison curve between the two-clustering methods k-means and
586 agglomerative (bottom-up) so that k-means showed excellent results better than the other
587 method.
588 Fig.17 gave a curve for the time consumed for each method to obtain these results, showing a
589 significant difference in the time taken between the two methods, with k-means saving time
590 while preserving its results better than the other method.
591 Generally, the k-means algorithm is easy to implement and more suitable with big data such as
592 SoS (Nasraoui and N'Cir 2019).

593 **Comparison between the k-means algorithm and some clustering algorithms**

594 The choice of the k-mean algorithm as a clustering technique for optimizing SSBFCSoS is
595 supported. SoS features are a combination of some IoT features and some of the big data,
596 therefore, we need a clustering algorithm with the following features for the fusion into
597 SSBFCSoS:
598   ▪ Dealing with huge datasets such as k-means, CURE, BIRCH and CHAMELEON (Sanse
599   and Sharma 2015), STING (Oyelade, Isewon et al. 2016), Agglomerative, and
600   HASTREAM (Nasraoui and N'Cir 2019). We display the data size criteria in the second
601   column Table 11.
602   ▪ Low time consumption. K-means is a very fast algorithm reducing the overall consumed
603   time (Nasraoui and N'Cir 2019).
604   ▪ Not requiring special hardware as all chosen algorithms in Table 11.
605   ▪ Scalability, the k-means algorithm is at the forefront of scalability as for SoS (Sholla,
606   Kaur et al. 2017).
607   ▪ Accepting overlapping clusters, which is the effect on conflict solution. k-means is one of
608   the algorithms that do not accept overlap between clusters (Khanmohammadi, Adibeig et
609   al. 2017) as well as OPTICS (Mirzaie, Barani et al. 2015), HASTREAM and Liar Tree
610   (Nasraoui and N'Cir 2019), While algorithms C-means (He, Guo et al. 2018), SOM
611   (Sarlin and Eklund 2011) and K-Medoids (Arora and Varshney 2016) accept overlapping.
612   This comparison is in column four, Table 11.

613   ▪   General implementation feasibility, as a trend in 2021 Artificial Intelligence utilises deep
614   learning in clustering big data but that need proper hardware and cloud system, and
615   applying to a certain application with the training dataset. K-means is easily applied to
616   any general SoS application, and can easily be applied to devices with available medium
617   capabilities as well. Finally, k-means is still one of the top five clustering algorithms
618   (Bangui, Ge et al. 2018).
619 For these reasons, the k-means algorithm was the base of our proposed algorithm SSBFCSoS.
620 We have compared k-means distance functions (Section 5), then adopted a more appropriate
621 specific measurement method as explained.
622

Conclusions

624 Increasing the SoS capacity to accommodate more systems results in some issues among the SoS
625 Components systems. Conflict is one of these issues. This paper focused on how to enhance the
626 speed of detection and solution of conflicts that may arise while integrating new systems into an
627 existing SoS.
628 We presented a method based on the use of the k-means clustering technique. Each cluster
629 contains nearby systems according to pre-specified criteria. We can consider each cluster a Sub
630 SoS (S-SoS) which in turn form the major SoS. We proposed the Smart Semantic Belief
631 Function Clustered System of Systems (SSBFCSoS) which is an enhancement of the Ontology
632 Belief Function System of Systems (OBFSoS). The proposed method proved the ability to detect
633 and solve conflicts.
634 In order to test the applicability of the SSBFCSoS and compare its performance with other
635 approaches, two well-known datasets were employed. They are (Glest & StarCraft Brood War).
636 With each dataset, 15 test cases were examined. Considering that each testing was after every ten
637 attempts of learning. To evaluate the proposed method, we applied it to homogeneous systems
638 cases. The Glest game for a sample of about 70 systems divided into three clusters (S-SoSs).
639 StarCraft Brood War was selected for a sample of about 100 systems divided into three clusters
640 (S-SoSs). Furthermore, we have picked these cases specifically for the application because we
641 intend to use homogeneous states.
642 Using clustering techniques showed better performance ratios. The proposed method
643 (SSBFCSoS) demonstrated the time-reducing effect of SoS activities with the early detection of
644 conflicts to be dealt with. We adopted an essential type of conflict, which is the struggle to
645 obtain the optimal Decision for SoS. As a learned system, it avoids similar conflicts during
646 attempts. It also helped to improve results and allowed us to deal with more CS in SoS with
647 saving much time.
648 Two types of clustering algorithms have been tried; k-means and agglomerative (bottom-up).
649 The algorithm k-means was adopted for SSBFCSoS for its better results, not only in terms of
650 resolving conflicts but for the less time, it took as well.

651   In many cases, clustering results enabled the SSBFCSoS to check the conflict state while still
652   handling conflict optimally. We achieved, on average, 89% in solving the conflict compared to
653   77% of the other well-known approaches. Moreover, it showed an acceleration of up to
654   proportionality over previous approaches for about 16% in solving conflicts as well. It also
655   reduces the frequency of the same conflicts by approximately 23% better than the other method,
656   not only in the same cluster but even while combining different clusters. Eventually; the positive
657   effect of the clustering process has appeared, the number of Component Systems (CS) has almost
658   tripled compared to other methods while preserving integration and conflict resolution.

659   ## Future Work

660   The evaluation of this approach appears promising, but scalability issues remain to addressed.
661   Consequently, future business trends should include developing more specific technologies to
662   define conflict for larger quantities of systems and rules of conflicts. Besides, dealing with
663   heterogeneous systems such as those found in smart cities. It can also be used as the
664   classification method developed for different types of conflicts to provide orderly results to the
665   user.
666   We used the k-means technique which depends on identifying the desired number of clusters
667   before initiating the clustering process. We found that when some new systems are added to the
668   already existing SoS, the system may belong to an S-SoS far from the one closest to it. That
669   deviates us from the idea that we relied on in classification to some extent. This would prompt us
670   to try different clustering techniques in the future.
671   Finally, the application on different numbers of systems within more than one method of
672   clustering shows another issue; which is how to reach the ideal number of systems within each
673   cluster.
674   Implementation showed that when the number of systems within one cluster reaches a specific
675   number, the SoS becomes ideal in achieving its goals, which is evidently, the ideal number of
676   clusters to achieve the SoS goal. This opens up another avenue for research on the same issue for
677   future work. Besides, we will put a competitive study with Artificial Intelligence clustering
678   techniques for the generalization of our proposed method (SSBFCSoS) in the future as well.
679

680   ## References

681   Agarwal, S., C. H. Dagli and L. E. Pape (2016). Computational intelligence based complex
682   adaptive system-of-system architecture evolution strategy. Complex Systems Design &
683   Management, Springer**: 119-132.
684   Ahmed, H., R. Salem and S. a. S. Saleh (2015). "Clustering Algorithm for Distributed Real-Time
685   Database sites." IJCI. International Journal of Computers and Information **4**(1): 11-20.
686   Arora, P. and S. Varshney (2016). "Analysis of k-means and k-medoids algorithm for big data."
687   Procedia Computer Science **78**: 507-512.
688   Bangui, H., M. Ge and B. Buhnova (2018). Exploring Big Data Clustering Algorithms for Internet
689   of Things Applications. IoTBDS.

690 Bodeau, D. J. (1994). System-of-systems security engineering. Tenth Annual Computer
691 Security Applications Conference, IEEE.
692 Boehm, B. and J. A. Lane (2007). "Using the incremental commitment model to integrate
693 system acquisition, systems engineering, and software engineering." CrossTalk **19**(10): 4-9.
694 Chiprianov, V., L. Gallon, M. Munier, P. Aniorte and V. Lalanne (2014). "The systems-of-
695 systems challenge in security engineering." Groupement De Recherche CNRS du Génie de la
696 Programmation et du Logiciel: 163.
697 Dimitriadis, V. K. (2009). "Reinforcement Learning in Real Time Strategy Games Case Study on
698 the Free Software Game Glest." Department of Electronic and Computer Engineering Technical
699 University of Crete. China.
700 Dor, S. (2014). "The heuristic circle of real-time strategy process: A starcraft: Brood war case
701 study." Game Studies **14**(1).
702 Engineering, D. o. S. (2010). "Systems Engineering Guide for System of
703 Systems." Defense Research and Engineering, Pentagon, Washington, D.C. 20301-3040: 17.
704 Garanina, N., E. Sidorova and S. Anokhin (2015). Conflict resolution in multi-agent systems with
705 typed connections for ontology population. International Andrei Ershov Memorial Conference on
706 Perspectives of System Informatics, Springer.
707 Gorod, A., B. E. White, V. Ireland, S. J. Gandhi and B. Sauser (2014). Case studies in system of
708 systems, enterprise systems, and complex systems engineering, CRC Press.
709 Gutierrez, R. (2018). A Requirement Ontology To Guide The Analysis Of System Life Cycle
710 Processes, Concordia University.
711 He, X., K. Guo, Q. Liao and Q. Yan (2018). Overlapping Community Detection Algorithm Based
712 on Spectral and Fuzzy C-Means Clustering. CCF Conference on Computer Supported
713 Cooperative Work and Social Computing, Springer.
714 Idrees, A. M. and W. H. Gomaa (2020) "A Proposed Method for Minimizing Mining Tasks' Data
715 Dimensionality."
716 Khanmohammadi, S., N. Adibeig and S. Shanehbandy (2017). "An improved overlapping k-
717 means clustering method for medical applications." Expert Systems with Applications **67**: 12-18.
718 Ki-Aries, D., S. Faily, H. Dogan and C. Williams (2018). Assessing system of systems security
719 risk and requirements with OASoSIS. 2018 IEEE 5th International Workshop on Evolving
720 Security & Privacy Requirements Engineering (ESPRE), IEEE.
721 Lane, J. A. and B. Boehm (2019). Systems of Systems Thinking. Systems Engineering in
722 Context, Springer: 553-564.
723 Luzeaux, D. (2014). SoS and large-scale complex systems architecting. Complex Systems
724 Design & Management, Springer: 39-49.
725 Madan, B. B. (2015). "System of Systems Security." Modeling and Simulation Support for
726 System of Systems Engineering Applications: 565.
727 Maier, M. W. (1998). "Architecting principles for systems-of-systems." Systems Engineering:
728 The Journal of the International Council on Systems Engineering **1**(4): 267-284.
729 Mirzaie, M., A. Barani, N. Nematbakkhsh and M. Beigi (2015). "OverDBC: A new density-based
730 clustering method with the ability of detecting overlapped clusters from gene expression data."
731 Intelligent Data Analysis **19**(6): 1311-1321.
732 Mokhtarpour, B. and J. Stracener (2014). Application of a clustering technique in identifying the
733 "best" System of Systems (SoS) during development. 2014 IEEE International Conference on
734 Systems, Man, and Cybernetics (SMC), IEEE.
735 Muflikhah, L., W. Widodo, W. F. Mahmudy and S. Solimun (2020) "A Support Vector Machine
736 Based on Kernel K-Means for Detecting the Liver Cancer Disease."
737 Nasraoui, O. and C.-E. B. N'Cir (2019). Clustering Methods for Big Data Analytics. Techniques,
738 Toolboxes and Applications, Springer: 192.

739 Oyelade, J., I. Isewon, F. Oladipupo, O. Aromolaran, E. Uwoghiren, F. Ameh, M. Achas and E.
740 Adebiyi (2016). "Clustering algorithms: their application to gene expression data." Bioinformatics
741 and Biology insights **10**: BBI. S38316.
742 Peculis, R. and F. Shirvani (2017). "Infrastructure system of systems integrity."
743 Popper, S. W., S. C. Bankes, R. Callaway and D. DeLaurentis (2004). "System of systems
744 symposium: Report on a summer conversation." Potomac Institute for Policy Studies, Arlington,
745 VA **320**.
746 Ramsbotham, O., H. Miall and T. Woodhouse (2011). Contemporary conflict resolution, Polity.
747 Robinson, W. N., S. D. Pawlowski and V. Volkov (2003). "Requirements interaction
748 management." ACM Computing Surveys (CSUR) **35**(2): 132-190.
749 Rokach, L. and O. Maimon (2005). Clustering methods. Data mining and knowledge discovery
750 handbook, Springer**:** 321-352.
751 Sage, A. P. (2003). Conflict and risk management in complex system of systems issues.
752 SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and
753 Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483), IEEE.
754 Sanse, K. and M. Sharma (2015). "Clustering methods for Big data analysis." International
755 Journal of Advanced Research in Computer Engineering & Technology (IJARCET) **4**(3).
756 Sarlin, P. and T. Eklund (2011). Fuzzy clustering of the self-organizing map: some applications
757 on financial time series. International Workshop on Self-Organizing Maps, Springer.
758 Sholla, S., S. Kaur, G. R. Begh, R. N. Mir and M. A. Chishti (2017). "Clustering Internet of
759 Things: A Review." Journal of Science and Technology: Issue on Information and
760 Communications Technology **3**(2): 21-27.
761 Singh, A., A. Yadav and A. Rana (2013). "K-means with Three different Distance Metrics."
762 International Journal of Computer Applications **67**(10).
763 Solanki, K. and P. Pittalia (2016). "Review of face recognition techniques." International Journal
764 of Computer Applications **133**(12): 20-24.
765 Spanoudakis, G. and A. Zisman (2001). Inconsistency management in software engineering:
766 Survey and open research issues. Handbook of Software Engineering and Knowledge
767 Engineering: Volume I: Fundamentals, World Scientific**:** 329-380.
768 Viana, T., A. Zisman and A. K. Bandara (2017). Identifying conflicting requirements in systems
769 of systems. 2017 IEEE 25th International Requirements Engineering Conference (RE), IEEE.
770 Wu, W., Z. Xu, G. Kou and Y. Shi (2020). "Decision-Making Support for the Evaluation of
771 Clustering Algorithms Based on MCDM." Complexity **2020**.
772 Younes, H. R., A. S. E. Ahmed and E. K. Elsayed "Enhancement Belief Function with Ontology
773 to Solve the Conflict for the System of Systems."
774

Box 1(on next page)

Algorithm 1 Smart Sematic Belief Function Clustered System of Systems (SSBFCSoS)

1. **Input**
   s_i $(s_1, s_2, ..., s_n)$ $(i = 2, ..., n)$ // Number of desired systems in each cluster
   C_l $(c_1, c_2, ..., c_r)$ $(l = 1, 2, ..., r)$ // Number of desired clusters
2. $C = 0$ // Cluster Index
3. $ClusteredSystems = 0$
4. for (each unclustered s_i)
5. $s_i \leftarrow RandomSystems$ (i, n) // Choose system at random
6. $C_l \leftarrow s_i$ // Make this random system as first cluster center
7. C ++ // initialize new cluster
8. $ClusteredSystems$ ++
9. **for** $l \leftarrow 2 ... r$ **do** // loop over the rest of the centres
10. **for** $i \leftarrow 1 ... n$ **do** // loop over the systems
11. $d_{s_i} \leftarrow \min_{l' < l} \| sqr(d_{s_i} - d_{C_{l_{cent}}}) \|^2$ // Compute the distance to the closest centre
12. Add s_i to the cluster C_l // Systems enter the closest centroid Clusters
13. **end for**
14. **for** $i \leftarrow 1 ... n$ **do** // loop over the systems again
15. $c_l \leftarrow d^2_n / \sum_i d^2_i$ // Compute a distribution proportional
16. **end for**
17. Recalculate centroids
18. $C_r \leftarrow s_n$ // Draw systems to each cluster from distance
19. **end for**
20. if $(ClusteredSystems = s_n)$
    break; // Clustering is complete
21. Implement OBFSoS Algorithm between generated clusters results // $OSoS_x$
22. **for** $l = 1 : r$ **do** // Clusters
23. Implement OBFSoS for each cluster
24. **end for**
25. Get output from OBFSoS X_{ij}, w_{ij} // Decision & Weight
26. $(X_{ij})_{new} \leftarrow X_{ij}$, $(w_{ij})_{new} \leftarrow w_{ij}$, $(X_{ij})_{old} \leftarrow 0$, $(w_{ij})_{old} \leftarrow 0$
27. **for** $x = 1, 3, 4, ..., r$ **do** // General SoS
28. if $(w_{ij})_{new} > (w_{ij})_{old}$
29. $GSoSX_{ij} \leftarrow (X_{ij})_{new}$
30. $(X_{ij})_{old} \leftarrow (X_{ij})_{new}$
31. else if $(w_{ij})_{old} > (w_{ij})_{new}$
32. $GSoSX_{ij} \leftarrow (X_{ij})_{old}$
33. else // $\because (w_{ij})_{old} = (w_{ij})_{new}$
34. $GSoSX_{ij} \leftarrow (X_{ij})_{old} \bowtie \leftarrow (X_{ij})_{new}$
35. $(X_{ij})_{old} \leftarrow (X_{ij})_{new}$
36. **end for**
37. A set of n systems for each cluster
38. **for** $i = 1 : n$ **do**
39. **for** $j = 1 : n$ **do**
40. $d(i,j) = $ distance function $(c(i), c(j))$
41. **end for**
42. $pair(i) = \min (d(i, j = 1 : n))$

43. ***end for***
44. ***for $k = 1{:}n$ do***
45. $MS_{SoS} = pair(i)$
46. *Get min distance* (MS_{SoS}, S_{SoS})
47. $MSoS = merge\ (MS_{SoS} \& S_{SoS})$
48. *back to step* 13
49. ***end for***
50. $GSoS = MSoS$

Output $GSoSX_{ij}$

Glest # of conflicts rate generated for SSBFCSoS vs. OBFSoS

Glest simulation attempts	# of conflicts	
	SSBFCSoS	OBFSoS
1	54	131
2	47	125
3	45	120
4	38	115
5	34	107
6	30	102
7	25	95
8	19	88
9	17	82
10	15	70

Glest Conflicts rate in decisions generated before and after implementing the SSBFCSoS vs. OBFSoS algorithms

Glest Simulation Attempts	SSBFCSoS								OBFSoS	
	Conflicts rate for systems n = 70, Run time = 300 seconds									
	Before Method				After Method				# conflicts Before Method	# conflicts After Method
	clusters conflicts in decision			# conflicts	clusters conflicts in decision			# conflicts		
	C1	C2	C3		C1	C2	C3			
1	0.250	0.350	0.400	**0.370**	0.200	0.300	0.300	**0.296**	0.397	0.366
2	0.250	0.375	0.375	**0.340**	0.188	0.313	0.375	**0.298**	0.408	0.360
3	0.267	0.400	0.333	**0.333**	0.133	0.267	0.267	**0.222**	0.417	0.350
4	0.333	0.333	0.333	**0.316**	0.167	0.250	0.250	**0.211**	0.417	0.339
5	0.273	0.273	0.455	**0.324**	0.091	0.182	0.364	**0.206**	0.411	0.346
6	0.222	0.222	0.556	**0.300**	0.111	0.111	0.333	**0.133**	0.422	0.343
7	0.500	0.167	0.333	**0.240**	0.333	0.167	0.167	**0.160**	0.421	0.326
8	0.250	0.250	0.500	**0.211**	0.250	0.000	0.250	**0.105**	0.398	0.341
9	0.667	0.000	0.333	**0.176**	0.333	0.000	0.000	**0.059**	0.415	0.341
10	0.333	0.333	0.333	**0.200**	0.000	0.000	0.333	**0.067**	0.414	0.357

Glest Conflicts rate in a design generated before and after implementing the SSBFCSoS vs. OBFSoS algorithms

Glest Simulation Attempts	SSBFCSoS								OBFSoS	
	Conflicts rate for systems n = 70, Run time = 300 seconds									
	Before Method				After Method					
	clusters conflicts in design			# conflicts	clusters conflicts in design			# conflicts	# conflicts Before Method	# conflicts After Method
	C1	C2	C3		C1	C2	C3			
1	0.308	0.231	0.462	**0.241**	0.231	0.154	0.308	**0.167**	0.282	0.260
2	0.273	0.273	0.455	**0.234**	0.182	0.182	0.364	**0.170**	0.280	0.264
3	0.250	0.417	0.333	**0.267**	0.167	0.250	0.250	**0.178**	0.275	0.250
4	0.300	0.400	0.300	**0.263**	0.200	0.200	0.200	**0.158**	0.278	0.243
5	0.222	0.444	0.333	**0.265**	0.111	0.333	0.111	**0.147**	0.290	0.252
6	0.375	0.375	0.250	**0.267**	0.375	0.125	0.125	**0.167**	0.284	0.235
7	0.286	0.571	0.143	**0.280**	0.143	0.286	0.000	**0.120**	0.305	0.242
8	0.000	0.333	0.667	**0.316**	0.000	0.167	0.167	**0.105**	0.295	0.250
9	0.400	0.600	0.000	**0.294**	0.200	0.200	0.000	**0.118**	0.293	0.244
10	0.250	0.500	0.250	**0.267**	0.000	0.250	0.000	**0.067**	0.286	0.243

Glest Conflicts rate in code generated before and after implementing the SSBFCSoS vs. OBFSoS algorithms

Glest Simulation Attempts	SSBFCSoS								OBFSoS	
	Conflicts rate for systems n = 70, Run time = 300 seconds									
	Before Method				After Method					
	clusters conflicts in code			# conflicts	clusters conflicts in code			# conflicts	# conflicts Before Method	# conflicts After Method
	C1	C2	C3		C1	C2	C3			
1	0.417	0.250	0.333	**0.222**	0.250	0.167	0.167	**0.130**	0.153	0.145
2	0.333	0.333	0.333	**0.255**	0.083	0.167	0.167	**0.106**	0.152	0.136
3	0.455	0.273	0.273	**0.244**	0.182	0.091	0.091	**0.089**	0.150	0.125
4	0.300	0.300	0.400	**0.263**	0.100	0.100	0.200	**0.105**	0.139	0.122
5	0.333	0.222	0.444	**0.265**	0.111	0.111	0.111	**0.088**	0.140	0.121
6	0.250	0.375	0.375	**0.267**	0.125	0.125	0.125	**0.100**	0.137	0.127
7	0.375	0.250	0.375	**0.320**	0.125	0.000	0.125	**0.080**	0.126	0.116
8	0.167	0.667	0.167	**0.316**	0.000	0.167	0.000	**0.053**	0.148	0.125
9	0.400	0.600	0.000	**0.294**	0.000	0.200	0.000	**0.059**	0.146	0.110
10	0.000	0.400	0.600	**0.333**	0.000	0.000	0.200	**0.067**	0.157	0.114

Glest Conflicts rate in security generated before and after implementing the SSBFCSoS vs. OBFSoS algorithms

Glest Simulation Attempts	SSBFCSoS							OBFSoS	
	Conflicts rate for systems n = 70, Run time = 300 seconds								
	Before Method				After Method			# conflicts Before Method	# conflicts After Method
	clusters conflicts in security			# conflicts	clusters conflicts in security			# conflicts		
	C1	C2	C3		C1	C2	C3			
1	0.444	0.222	0.333	**0.167**	0.333	0.111	0.222	**0.111**	0.168	0.153
2	0.375	0.375	0.250	**0.170**	0.250	0.250	0.125	**0.106**	0.160	0.152
3	0.286	0.286	0.429	**0.156**	0.143	0.143	0.286	**0.089**	0.158	0.142
4	0.500	0.500	0.000	**0.158**	0.333	0.333	0.000	**0.105**	0.165	0.130
5	0.400	0.200	0.400	**0.147**	0.200	0.200	0.200	**0.088**	0.159	0.131
6	0.200	0.400	0.400	**0.167**	0.200	0.200	0.000	**0.067**	0.157	0.127
7	0.750	0.250	0.000	**0.160**	0.250	0.250	0.000	**0.080**	0.147	0.126
8	0.000	0.667	0.333	**0.158**	0.000	0.333	0.000	**0.053**	0.159	0.125
9	0.500	0.250	0.250	**0.235**	0.250	0.000	0.000	**0.059**	0.146	0.122
10	0.667	0.000	0.333	**0.200**	0.000	0.000	0.000	**0.067**	0.143	0.114

Glest # of resolved conflicts rate generated SSBFCSoS vs. OBFSoS with consumed time

Glest simulation attempts	SSBFCSoS		OBFSoS	
	Rate	Time	Rate	Time
1	0.296	281.11	0.076	300.00
2	0.319	270.59	0.088	300.00
3	0.422	262.28	0.133	288.89
4	0.421	214.67	0.165	278.26
5	0.471	204.44	0.15	283.49
6	0.533	180.39	0.167	277.78
7	0.560	161.40	0.189	270.18
8	0.684	161.85	0.159	280.30
9	0.706	122.34	0.183	272.36
10	0.733	113.58	0.171	276.19

StarCraft Brood War Conflicts rate generated for SSBFCSoS vs. OBFSoS

StarCraft Brood War simulation attempts	SSBFCSoS				OBFSoS			
	Conflicts rate for systems n = 100, Run time = 400 seconds							
	conflicts ratio				conflicts ratio			
	decision	design	security	code	decision	design	security	code
1	0.343	0.239	0.209	0.209	0.364	0.283	0.192	0.162
2	0.322	0.254	0.203	0.220	0.385	0.275	0.181	0.159
3	0.314	0.255	0.216	0.216	0.390	0.273	0.174	0.163
4	0.319	0.255	0.191	0.234	0.383	0.272	0.185	0.160
5	0.317	0.268	0.195	0.220	0.396	0.268	0.181	0.154
6	0.303	0.242	0.212	0.242	0.396	0.273	0.180	0.151
7	0.333	0.222	0.222	0.222	0.398	0.273	0.172	0.156
8	0.333	0.250	0.167	0.250	0.435	0.287	0.204	0.074
9	0.250	0.300	0.200	0.250	0.394	0.279	0.173	0.154
10	0.278	0.278	0.222	0.222	0.402	0.278	0.186	0.134

StarCraft Brood War Resolved Conflicts generated after implementing the SSBFCSoS vs. OBFSoS

StarCraft Brood War simulation attempts	SSBFCSoS				OBFSoS			
	Conflicts rate for systems n = 100, Run time = 400 seconds							
	Resolved conflicts ratio				Resolved conflicts ratio			
	decision	design	security	code	decision	design	security	code
1	0.269	0.224	0.164	0.149	0.343	0.258	0.177	0.157
2	0.254	0.220	0.169	0.153	0.352	0.264	0.181	0.159
3	0.255	0.235	0.157	0.157	0.355	0.250	0.163	0.163
4	0.255	0.234	0.149	0.149	0.352	0.241	0.167	0.167
5	0.268	0.195	0.146	0.122	0.362	0.248	0.161	0.161
6	0.212	0.182	0.152	0.121	0.367	0.245	0.158	0.158
7	0.222	0.185	0.148	0.111	0.375	0.242	0.156	0.156
8	0.208	0.208	0.125	0.083	0.389	0.259	0.176	0.167
9	0.100	0.200	0.100	0.100	0.365	0.240	0.154	0.163
10	0.056	0.056	0.111	0.111	0.309	0.227	0.155	0.144

StarCraft Brood War Resolved Conflicts rate generated after implementing the SSBFCSoS vs. OBFSoS with consumed time

StarCraft Brood War simulation attempts	SSBFCSoS		OBFSoS	
	Rate	Time	Rate	Time
1	0.194	369.15	0.066	399.28
2	0.203	251.26	0.044	377.43
3	0.196	198.5	0.070	370.44
4	0.213	189.34	0.074	355.31
5	0.268	158.55	0.067	341.37
6	0.333	131.51	0.072	335.58
7	0.333	127.28	0.070	317.08
8	0.375	101.45	0.009	304.21
9	0.500	98.56	0.077	299.25
10	0.667	87.54	0.165	289.11

StarCraft Brood War Resolving Conflicts rate generated after implementing the SSBFCSoS method using k-means clustering vs. Agglomerative (bottom-up) with consumed time

# systems	k-means		Agglomerative (bottom-up)	
	Rate	Time	Rate	Time
20	0.846	60.42	0.692	99.52
35	0.778	99.38	0.704	161.55
65	0.895	121.05	0.667	232.32
80	0.855	196.57	0.696	333.33
95	0.884	241.47	0.663	423.45
110	0.899	279.42	0.687	789.50
130	0.888	329.36	0.664	953.36
140	0.876	373.16	0.664	1696.17
170	0.884	406.38	0.661	1900.53
190	0.891	539.47	0.652	2884.42

Comparison between the k-means algorithm and some clustering algorithms

Algorithm	Dataset size	Scalability	Overlapping Clusters (Conflicts)	Time Consuming	Adapting with SoS/IoT
k-means	Huge	Yes	No	Low	Yes
K-medoids	Small	No	Yes	High	No
CLARA	Huge	Yes	No	Medium	Yes
Agglomerative (bottom-up)	Huge	Yes	No	High	Yes
STING	Huge	No	-	Medium	Yes
c-means	Medium	Yes	Yes	High	No
BIRCH	Huge	Yes	-	High	No
CURE	Huge	Yes	-	High	Yes
CHAMELEON	Huge	Yes	No	High	No
OPTICS	Small	No	No	Medium	Yes
HASTREAM	Huge	Yes	No	High	No
Liar Tree	Small	No	No	High	No
SOM	Small	No	Yes	High	Yes

Figure 1

The proposed Method
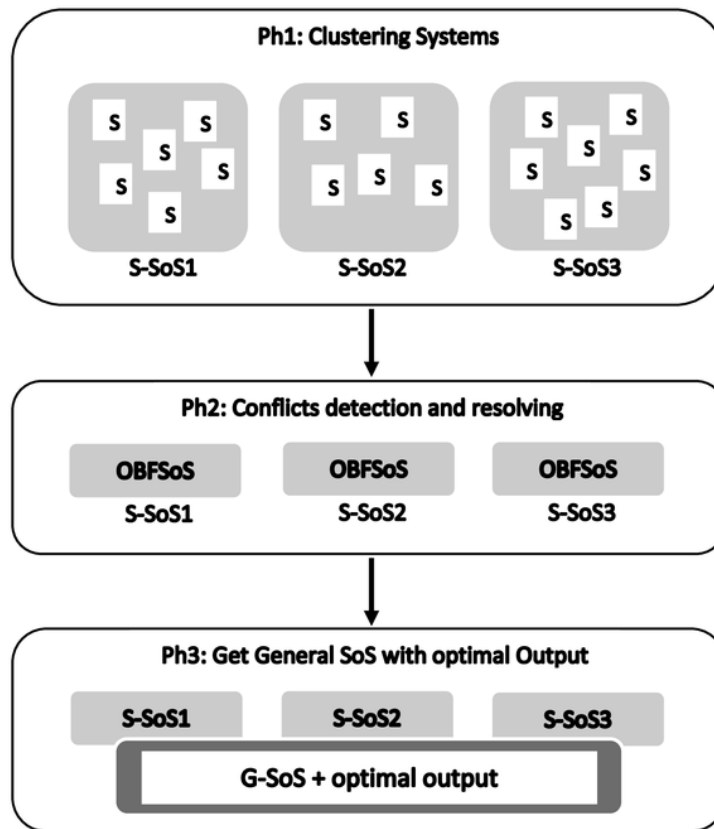
Figure 2

Conflicts rate in Glest for SSBFCSoS vs. OBFSoS
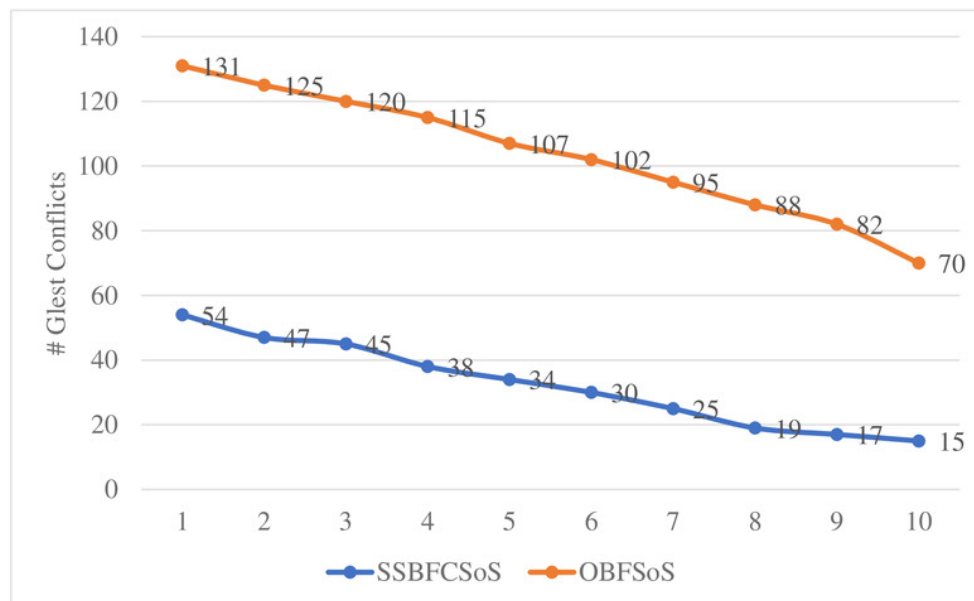
Figure 3

Glest Conflicts rate in decisions after implementing SSBFCSoS vs. OBFSoS
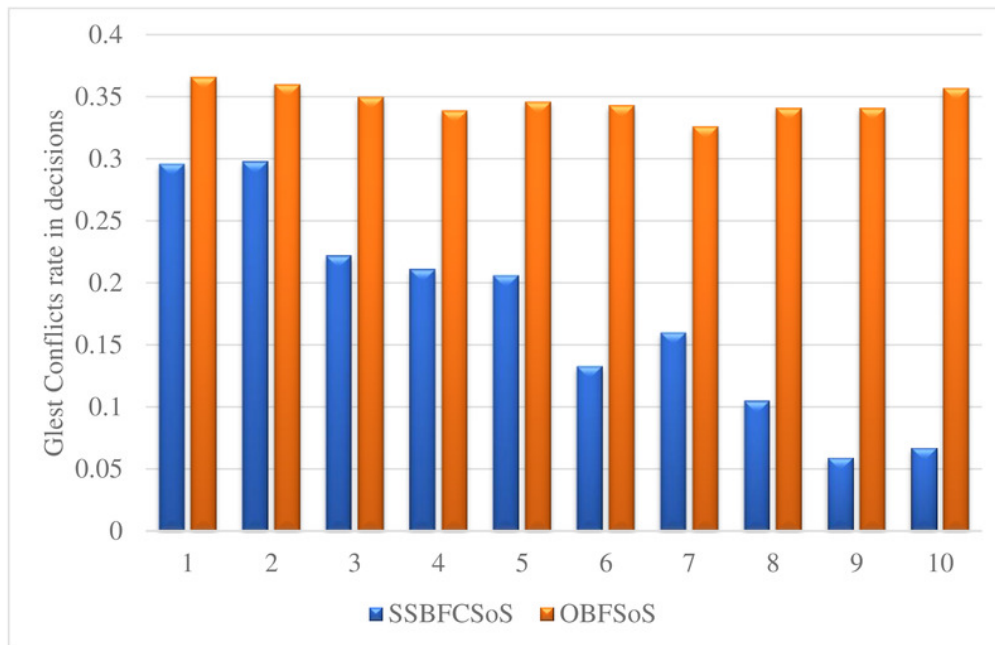
Figure 4

Glest Conflicts rate in design after implementing SSBFCSoS vs. OBFSoS
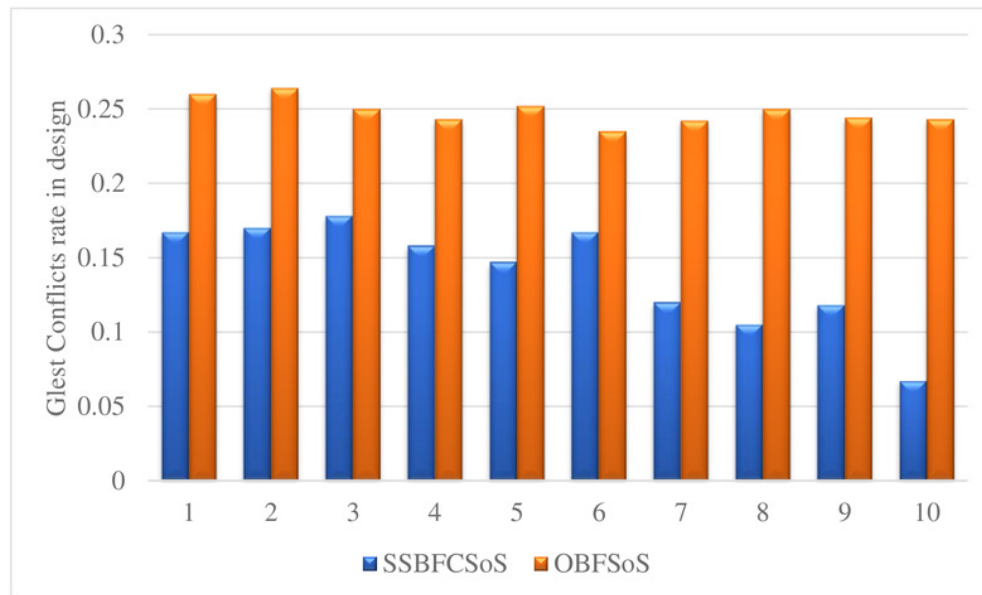
Figure 5

Glest Conflicts rate in code after implementing SSBFCSoS vs. OBFSoS
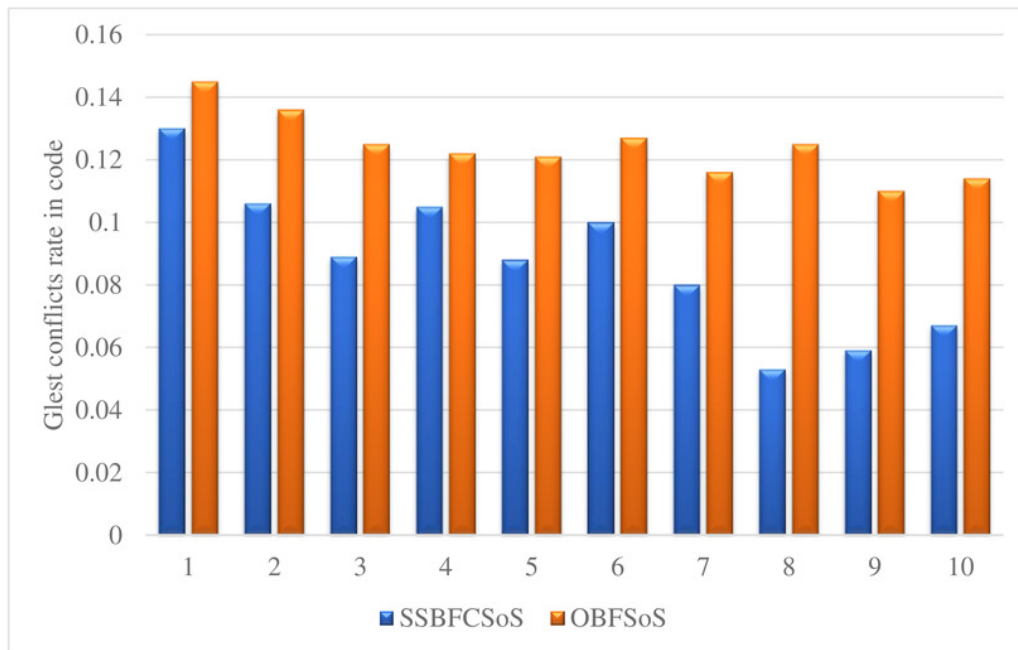
Figure 6

Glest Conflicts rate in security after implementing SSBFCSoS vs. OBFSoS
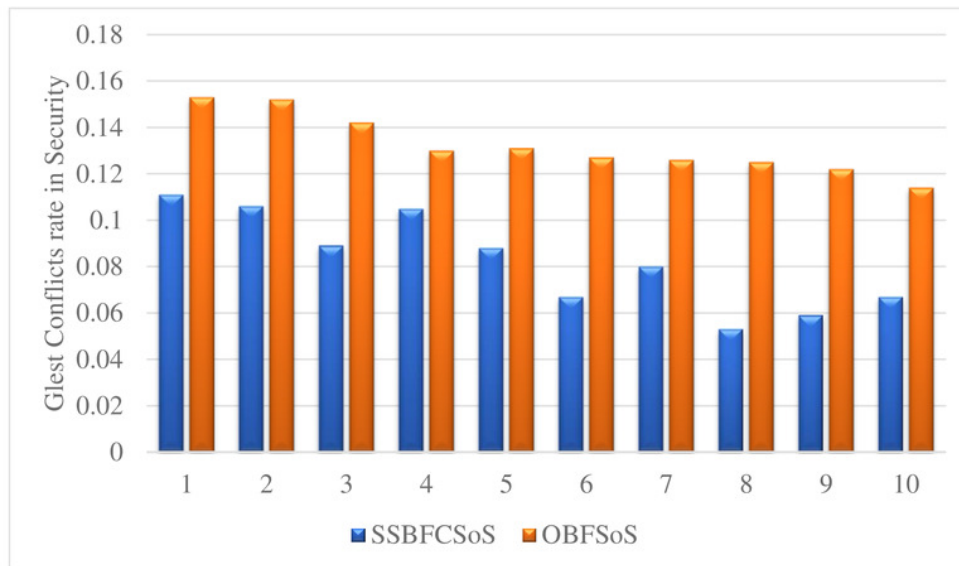
Figure 7

Glest Resolved Conflicts rate after implementing SSBFCSoS vs. OBFSoS

	1	2	3	4	5	6	7	8	9	10
SSBFCSoS	0.296	0.319	0.422	0.421	0.471	0.533	0.56	0.684	0.706	0.733
OBFSoS	0.076	0.088	0.133	0.165	0.15	0.167	0.189	0.159	0.183	0.171
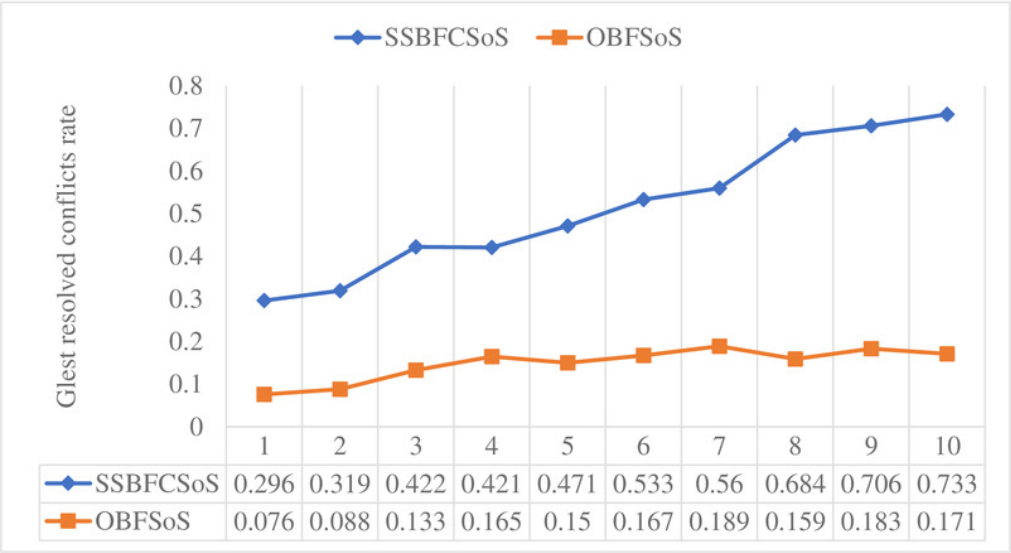
Figure 8

Glest Time Consumed in Conflicts Resolving during implementing SSBFCSoS vs. OBFSoS
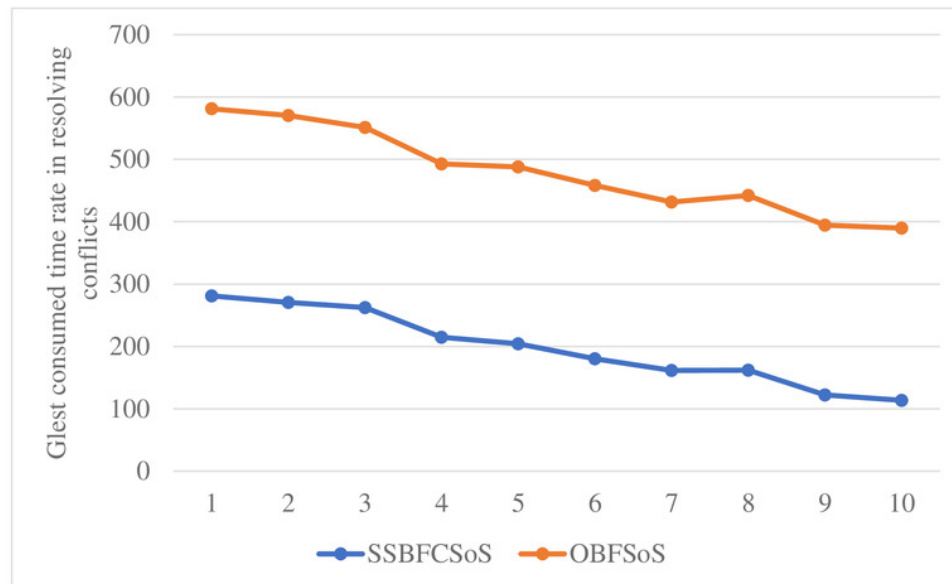
Figure 9

StarCraft Brood War Conflicts rate in decisions after implementing SSBFCSoS vs. OBFSoS
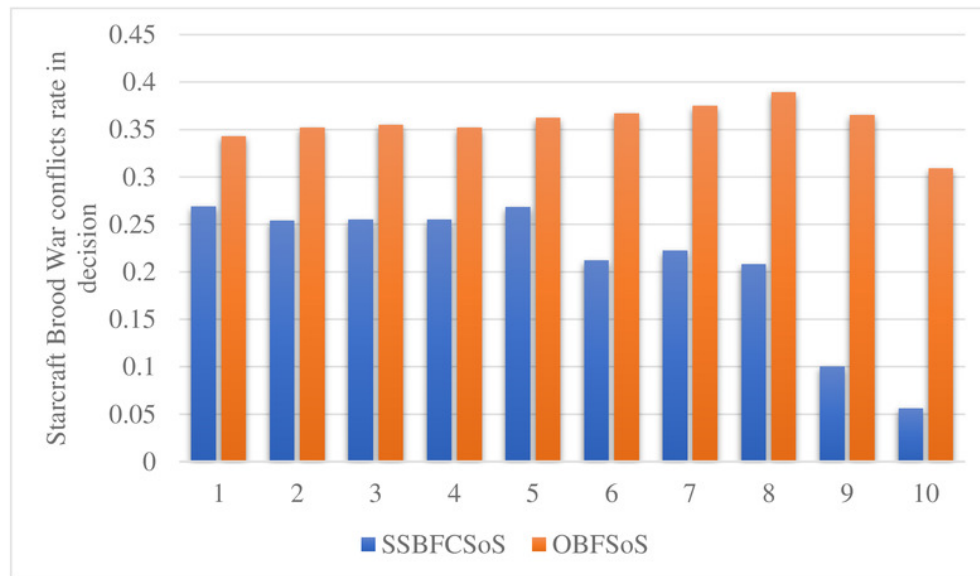
Figure 10

StarCraft Brood War Conflicts rate in design after implementing SSBFCSoS vs. OBFSoS

Figure 11

StarCraft Brood War Conflicts rate in security after implementing SSBFCSoS vs. OBFSoS
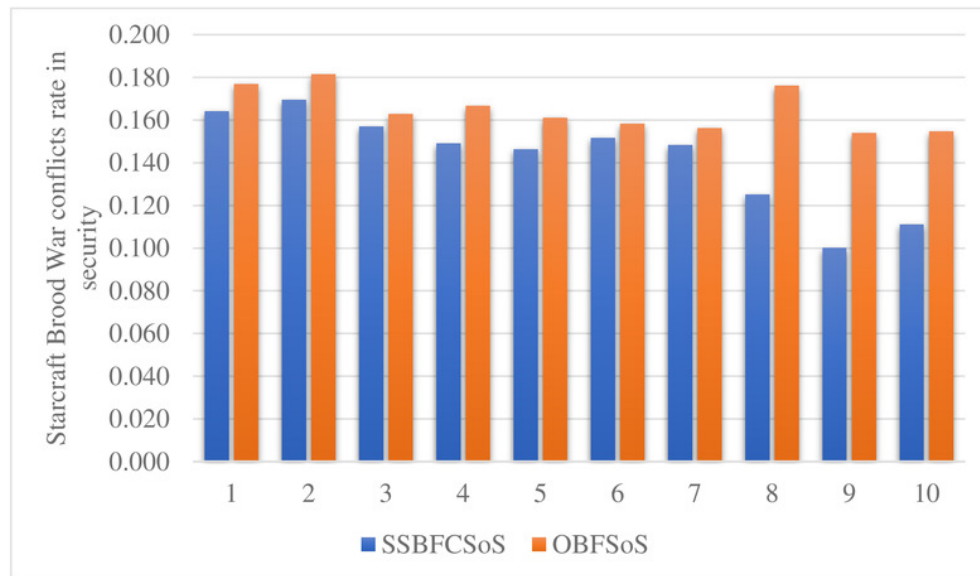
Figure 12

StarCraft Brood War Conflicts rate in code after implementing SSBFCSoS vs. OBFSoS
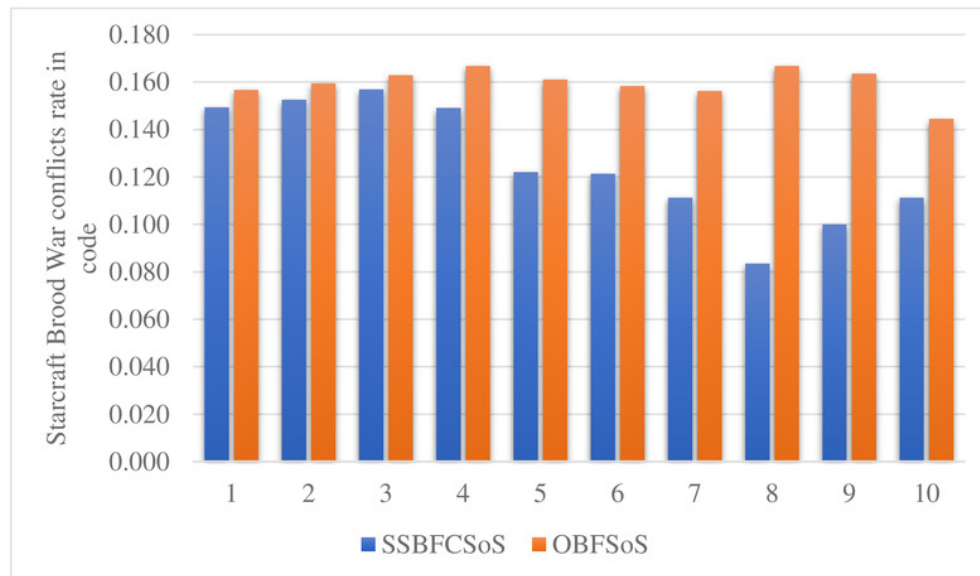
Figure 13

StarCraft Brood War Resolved Conflicts rate after implementing SSBFCSoS vs. OBFSoS

	1	2	3	4	5	6	7	8	9	10
SSBFCSoS	0.194	0.203	0.196	0.213	0.268	0.333	0.333	0.375	0.500	0.667
OBFSoS	0.066	0.044	0.070	0.074	0.067	0.072	0.070	0.009	0.077	0.165
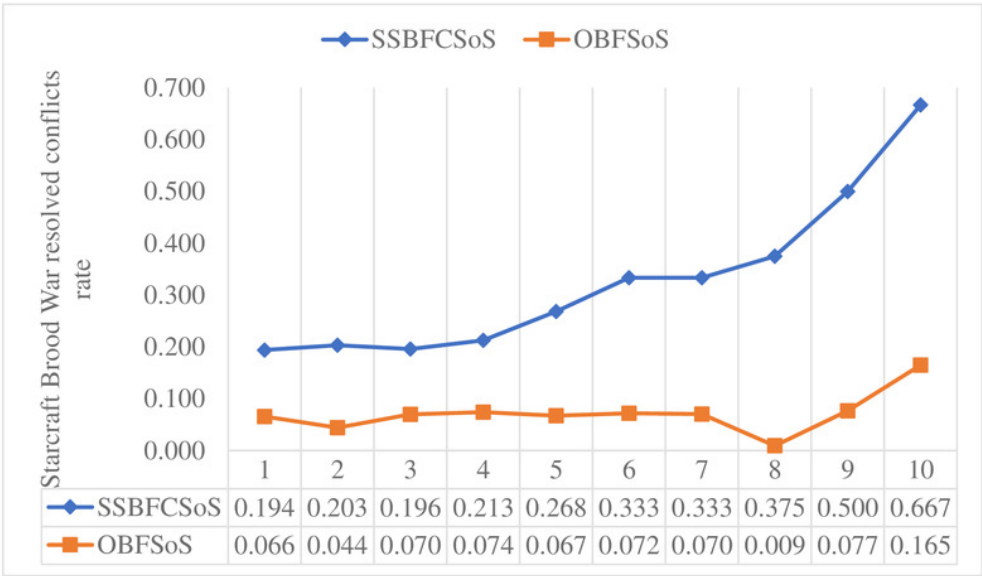
Figure 14

StarCraft Brood War Time Consumed in Conflicts Resolving during implementing
SSBFCSoS vs. OBFSoS
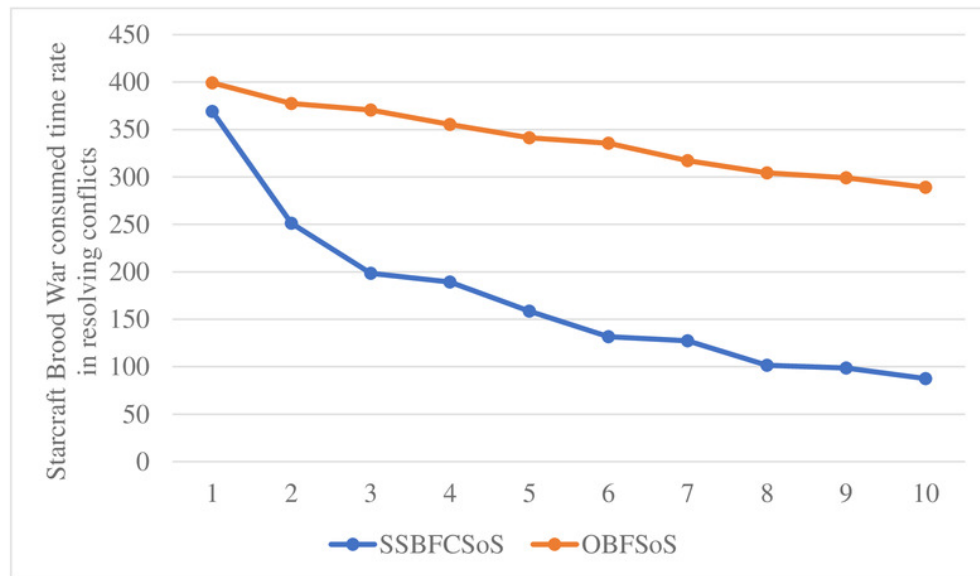
Figure 15

Resolved Conflicts rate for SSBFCSoS with different cluster numbers and systems
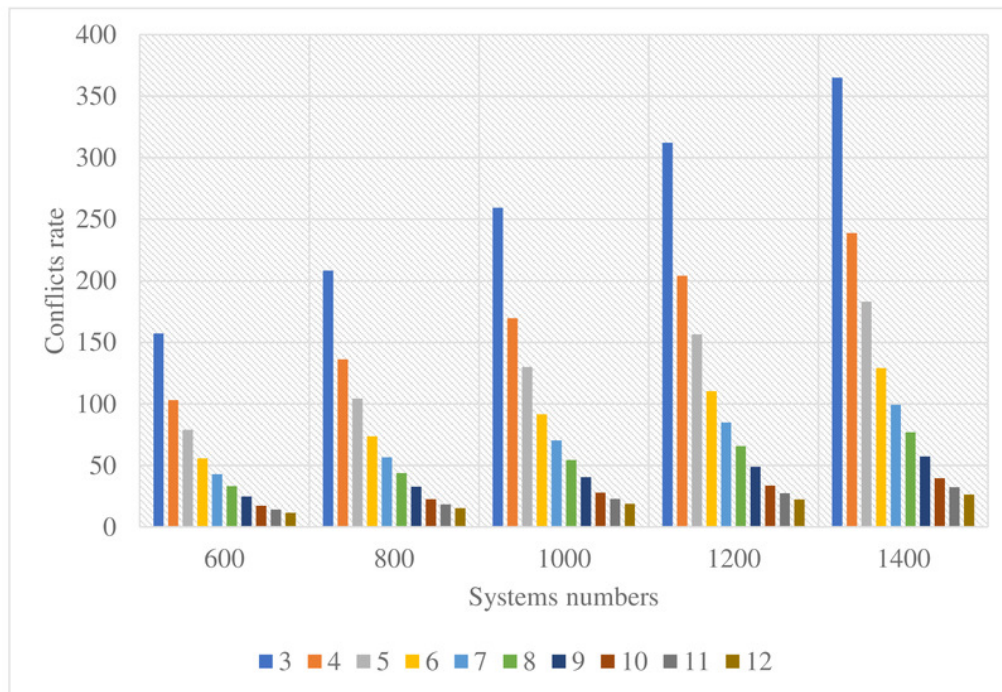
Figure 16

StarCraft Brood War Resolved Conflicts rate generated implementing the SSBFCSoS using k-means clustering vs. Agglomerative (bottom-up)
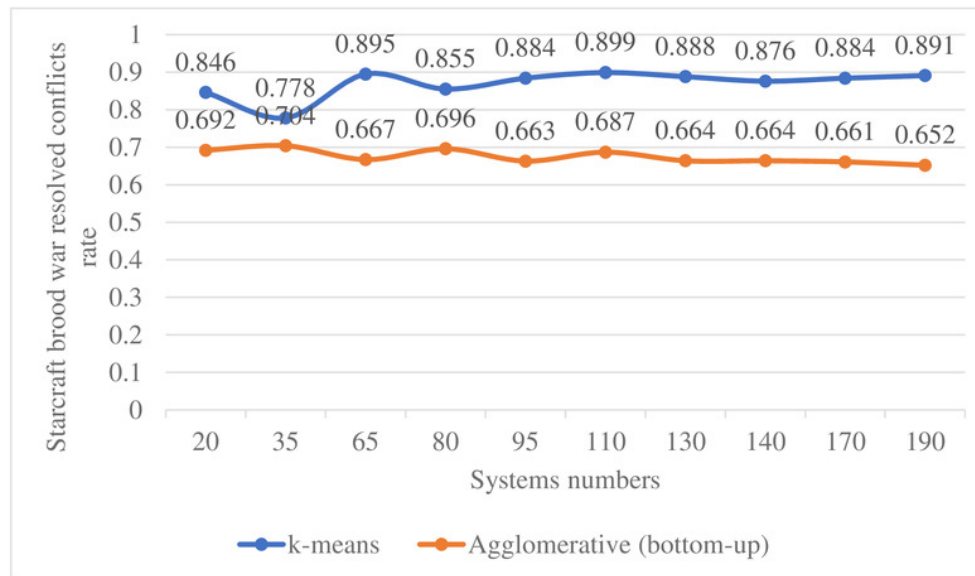
Figure 17

StarCraft Brood War Time Consumed in Conflicts Resolving during implementing SSBFCSoS using k-means vs. Agglomerative (bottom-up) clustering techniques