Supplemetary material for the paper 'Summary Generation for Figures in Biomedical Literature using Multiobjective Optimization'

Naveen Saini, Sriparna Saha, Veda Vikas Potnuru, Rahul Grover, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Institute of Technology Patna, Bihar, India-801103
Email: {naveen.pcs16, sriparna, potnuru.cs16, rahul.cs17, pb}@iitp.ac.in

1. Background knowledge

1.0.1. Self-organizing Map. SOM [1] is a special type of feed-forward artificial neural network consisting of two layers: input and output layer. In input layer, there are set of input vectors, while, in output layer a set of neurons are there usually arranged in a 2D grid. The principle of SOM states that the input vectors/patterns which are similar in input space come close to each other in the output space using the unsupervised training procedure. This principle is shown in Figure 1. For example, the input pattern's features corresponding to orange color are mapping to nearby neurons. Thus, help in clustering analysis of high dimensional data. It can also be used for mapping high dimensional input data to low dimensional space (output space).

The basic architecture of SOM is shown in Figure 1. Each neuron has a pre-defined position vector in the grid and weight vector of the same length as of input vector. For example, if $T = \{x_1, x_2....x_H\}$ is the set of input vectors (each on n-dimensional), then each neuron has a weight vector, $w^u = [w_1^u, w_2^u, \ldots, w_n^u]$ which is also n-dimensional and a position vector, z^u=(z_1^u, z_2^u) which is 2-dimensional (as neurons are arranged in a 2-D grid). Here 'u' is the index of a neuron. For our task, we follow the sequential learning approach to train the SOM discussed in [2].

Training of SOM starts after initializing initial learning rate (η_0), initial neighborhood radius (σ_0) and assigning each neuron a weight vector. In every iteration, a random input vector is selected followed by finding it's closest (winning) neuron using shortest Euclidean criteria. Then neighboring neurons of the closest neuron are identified using the position vectors of winning and other neurons. Here, a neighboring neuron will be that neuron whose Euclidean distance to winning neuron falls below a neighborhood radius (σ) in that iteration. Note that σ is continuously decreasing over the iteration and is defined as

$$\sigma = \sigma_0 \times \big(1 - \frac{current\_Iteration}{Maximum\_Iteration}\big) \tag{1}$$

Finally, updating the weight vector of the winning neuron and neighboring neurons takes place to bring them close to the presented input vector as

$$w^u = w^u + \eta * \exp(- \parallel z^u - z^{u'} \parallel_2) * (x - w^u) \tag{2}$$

where, u is the index of neighboring/winning neuron, η, which is defined as

$$\eta = \eta_0 \times \big(1 - \frac{current\_Iteration}{Maximum\_Iteration}\big) \tag{3}$$

is the learning rate decreasing over iterations, x is the input vector. This will continue until we reach maximum number of iterations.

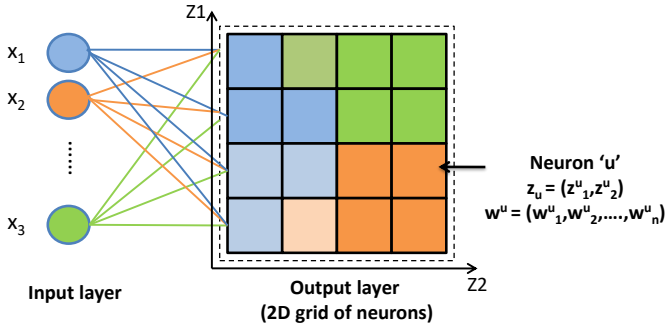


Figure 1. SOM Architecture. Here $x_p = \{x_p^1, x_p^2, \ldots, x_p^n\}$ is the input vector, $Z1$ and $Z2$ denote the axes of 2-d Map.

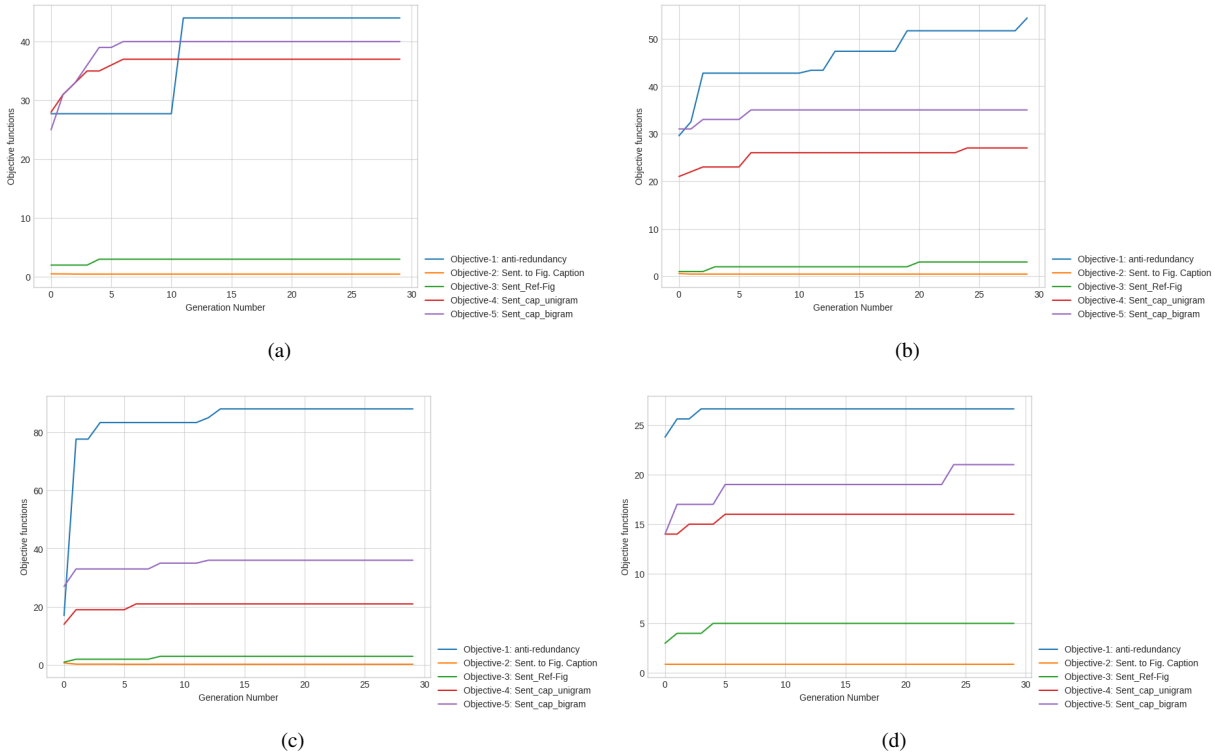


Figure 2. Sub-figures (a)-(d) shows the generation wise maximum different objectives functions score value used in our framework. These sub-figures corresponding to randomly chosen four figures to be summarized belonging to different biomedical articles.

1.1. A Binary Differential Evolutionary Algorithm for Optimization

Differential Evolution (DE) is a population-based global optimization technique proposed by Storn and Price [3] to solve real-world problems. There exist many variants of the DE; each differs in representation (real-coded or binary-coded) of the solution, new solution generation strategy and in the use of parameters. In our paper, a binary differential evolution algorithm [4] is used where each solution is represented as a binary vector. Each solution is associated with two or more objective functions in DE framework for multi-objective optimization. It executes similar to any other evolutionary algorithm. It starts with a set of solutions called as population. At time stamp (generation) t, ith solution is represented as

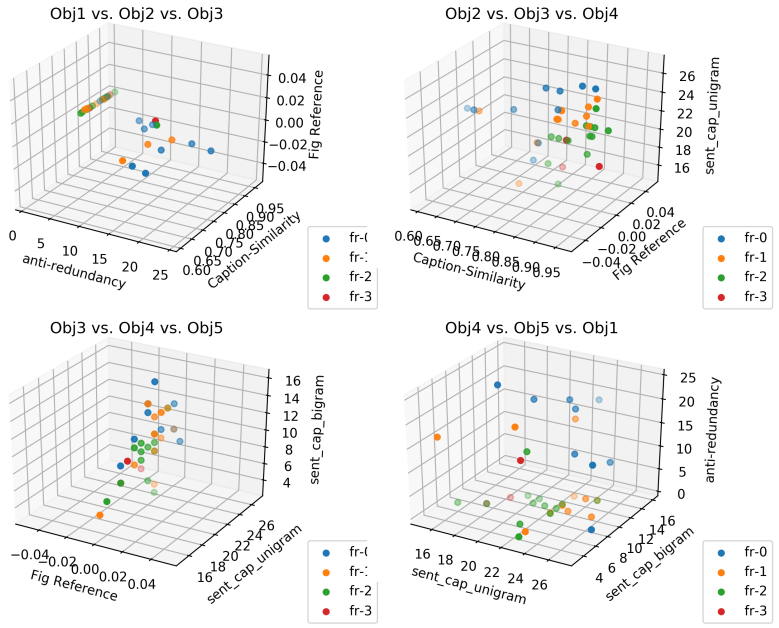
$$\vec{x}_i(t) = [x_{i,1}(t), x_{i,2}(t), \ldots, x_{i,n}(t)] \tag{4}$$

where, n is the length of the solution and $i = 1, 2 \ldots, \mid P \mid$, $\mid P \mid$ is the size of population, $x_{i,m}$ can take value either 0 or 1 for $m = 1, 2 \ldots, n$. For each current solution i, offspring y' is generated using crossover and mutation operations [3] [5] and then it undergoes evaluation in comparison with current solution, i. Crossover is the exchange of components between two solutions and mutation is the modification in the component. Only the better solution in terms of objective function values out of these two solutions (current and new offspring) can survive into next generation. More details about used binary DE can be found in the paper [4].

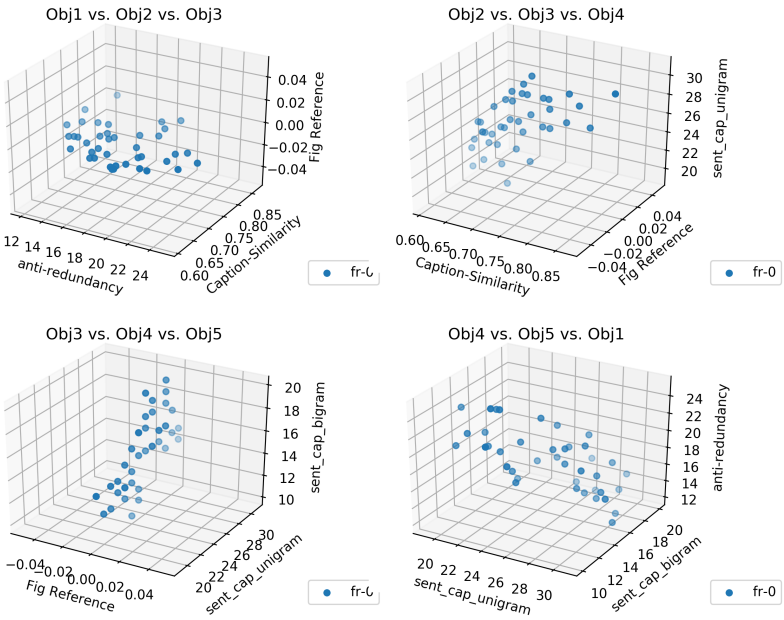
2. Results and discussion

2.1. Generation wise maximum objective functions score

As our algorithm is based on optimization, therefore, to check that, objective functions score are improving or not over the generation (all functions are of maximization type), we have plotted their generation wise maximum score obtained. In Fig 2, four sub-figures are shown which are corresponding to randomly chosen 4 figures to be summarized. From these figures, it can be evident that the all used objective functions are increasing over the generation and become constant after a particular generation. For example, the functions namely, number of sentences referring to that figure (SRTF) and sentence similarity with figure's caption (SSWFC) becomes constant after a fixed number of iteration as there can limited number of sentences in the article referring to the figure and having high similarity with the figure's caption of that figure. Anti-redundancy is seems to highly varying as there can be lot of sentences in the article.



(a)



(b)

Figure 3. Sub-figures (a) and (b) shows the Pareto fronts obtained after $0th$ and $25th$ generation of our proposed approach. These sub-figures corresponding to one randomly chosen figure to be summarized. The axes denote objective functions score value.

2.2. Pareto Fronts Obtained

Pareto fronts obtained by our proposed approach (MOOFigSum) corresponding to the best results are shown in sub-plots under Fig. 3. Here, coloured bullets represents the solutions in the objective space. These sub-plots are obtained after application of the proposed algorithm in the $0th$ and $25th$ generations on a randomly chosen single figure to be summarized. In the $0th$ generations, solutions are randomly distributed as shown in Fig. 3(a). As the number of generations passes, we will get a set of optimized solutions (as shown in Fig. 3(b)). In these sub-plots, fr-0, fr-1, fr-2 and fr-3, mentioned in the legend denotes the rank-1, rank-2, rank-3 and rank-4 solutions. Note that the best result is obtained after optimizing five

TABLE 1. SENSITIVITY ANALYSIS ON THE PARAMETERS USED BY THE PROPOSED ALGORITHM

S.No.	b	CR	F	H	Fig-1	Fig-2	Fig-3	Fig-4	Fig-5	Fig-6	Fig-7	Fig-8	Fig-9	Mean
1	5	0.2	0.8	4	0.36	0.75	0.40	0.60	0.52	0.49	0.40	0.57	0.37	0.50
2	5	0.6	0.2	5	0.40	0.75	0.40	0.50	0.48	0.43	0.40	0.57	0.40	0.48
3	5	0.8	0.5	6	0.36	0.86	0.50	0.67	0.54	0.44	0.53	0.57	0.37	0.54
4	6	0.2	0.2	6	0.36	0.67	0.40	0.67	0.52	0.42	0.13	0.29	0.37	0.43
5	6	0.6	0.5	4	0.36	0.86	0.50	0.67	0.59	0.50	0.40	0.57	0.42	0.54
6	**6**	**0.8**	**0.8**	**5**	0.50	0.86	0.50	0.67	0.62	0.55	0.53	0.57	0.41	**0.58**
7	7	0.2	0.5	5	0.40	0.67	0.40	0.44	0.57	0.41	0.40	0.57	0.38	0.47
8	7	0.6	0.8	6	0.33	0.86	0.29	0.67	0.57	0.47	0.40	0.57	0.42	0.51
9	7	0.8	0.2	4	0.44	0.75	0.40	0.67	0.59	0.47	0.40	0.57	0.43	0.52

objective functions. Therefore, it difficult to visualize all objective functions score in a single plot. However, we have taken three objective functions for consideration to draw the plots.

2.3. Sensitivity analysis on the Parameters Used

To evaluate the performance on the different parameters used by our proposed system, we have performed a sensitivity analysis. We have considered different values of the parameters (b, CR, F, and H) and shown in Table 1 based on the literate survey [4], [6], [7]. The parameters population size and maximum number of generation are excluded because more will be their value, more will the running time. Corresponding results are also reported in the same table. Here, all parameters are the differential evolution parameters. We have chosen only DE parameters because the quality of new solutions generated at each generation depends on parameters values [6]. These new solutions may help in reaching towards the global optimum solution in search space. We have chosen randomly 9 figures to be summarized with their associated text and ran our proposed approach with SOM-based operator and optimizing five objectives. In the Table1, the values under the columns 5 to 13 report the F-measure obtained. The $15th$ column reports the average value of F-measure overall figures under particular values of the parameters. The best average F-measure value of 0.58 was obtained at the parameter's value of 6, 0.8, 0.8, 5 corresponding to b, CR, F, and H, respectively. We are assuming that these parameters also work when we use our method with normal DE operators and with less number of objective functions. Therefore, these parameters values are used to perform all experiments in the main manuscript. Sub-figures (a) and (b) shows the Pareto fronts obtained after $0th$ and $25th$ generation of our proposed approach. These sub-figures corresponding to one randomly chosen figure to be summarized. The axes denote objective functions score value.

2.4. Qualities of Solutions Obtained in terms of F-measure

References

[1] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1, pp. 1–6, 1998.

[2] H. Zhang, A. Zhou, S. Song, Q. Zhang, X. Z. Gao, and J. Zhang, "A self-organizing multiobjective evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 792–806, Oct 2016.

[3] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[4] L. Wang, X. Fu, M. I. Menhas, and M. Fei, "A modified binary differential evolution algorithm," in *Life System Modeling and Intelligent Computing*. Springer, 2010, pp. 49–57.

[5] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[6] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.

[7] N. Saini, S. Saha, A. Jangra, and P. Bhattacharyya, "Extractive single document summarization using multi-objective optimization: Exploring self-organized differential evolution, grey wolf optimizer and water cycle algorithm," *Knowledge-Based Systems*, vol. 164, pp. 45–67, 2019.