Using a Social Media Inspired Optimization Algorithm to Solve the Set Covering Problem

Broderick Crawford[1(✉)], Ricardo Soto[1], Guillermo Cabrera[1], Agustín Salas-Fernández[2], and Fernando Paredes[3]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
{broderick.crawford,ricardo.soto}@pucv.cl
[2] Universidad Tecnológica de Chile INACAP, Santiago, Chile
jsalasf@inacap.cl
[3] Universidad Diego Portales, Santiago, Chile
fernando.paredes@udp.cl

Abstract. Currently, researchers have focused on solving large-scale and non-linear optimization problems. Metaheuristics as its prefix indicates, are superior heuristics that aim to deliver acceptable results to optimization problems in a short period of time, trying to achieve a correct balance between exploration and exploitation in the search for solutions. In this paper we present the application of a metaheuristic technique called Social media optimization algorithm for the resolution of the Set Covering Problem (SCP). This technique is inspired by the behavior of users of social networking platforms such as Twitter. The users through different interactions manage to make a *Tweet* more relevant than others. The user who generates the best Tweet, is recognized as a *celebrity*. This process of social relationship is precisely what allows us to find better solutions given the experiments and results presented in this document.

Keywords: Twitter Optimization · Social media · Metaheuristics · SCP

1 Introduction

Finding an optimal solution for an optimization problem is often a very challenging task [1], depending on the choice and the correct use of the correct algorithm. The choice of an algorithm may depend on the type of problem, the availability of algorithms, computational resources and time constraints. For large-scale, non-linear global optimization problems, there is often no agreed guideline for the choice of algorithm and, in many cases, there is no efficient algorithm. Several algorithms can be used to solve optimization problems. The conventional or classical algorithms are mostly deterministic.

G. Meiselwitz (Ed.): HCII 2019, LNCS 11578, pp. 43–52, 2019.
https://doi.org/10.1007/978-3-030-21902-4_4

Metaheuristics is another major field devoted to solving optimization problems. They are useful when finding the best solution is computationally very expensive. The key is to provide a way of finding a *good-enough* solution in a fixed amount of time. Metaheuristics, as the suffix says, are upper level heuristics. They are intelligent strategies to design or improve general heuristic procedures with high performance. In their original definition, metaheuristics are general purpose approximated optimization algorithms; they find a good solution for the problem in a reasonable time (not necessarily the optimal solution). They are iterative procedures that smartly guide a subordinate heuristic, combining different concepts to suitably explore and operate the search space.

Over time, these methods have also come to include any procedures that employ strategies for overcoming the trap of local optimality in complex solution spaces, especially those procedures that utilize one or more neighborhood structures as a mean for defining admissible transitions from one solution to another, or to transform solutions in a constructive process [2]. To get good solutions, any search algorithm must establish an adequate balance between two overlayed process characteristics:

Intensity is a mechanism that explores more throughly the portions of the search space that seem promising in order to make sure that the best solution in these areas is indeed found (space exploitation).

Diversity is a mechanism that forces the search into unexplored areas of the search space (space exploration). This balance is needed to quickly identify regions with good quality solutions and do not waste time in promising or visited regions. The metaheuristics are categorized in: Constructive heuristics: they start from an empty solution and continue adding components until a solution is built. (i.e.: GRASP [3], Ant Colony Optimization [4] and Intelligent Water Drops [5]). Trajectory methods: they start from an initial solution and then, iteratively, try to replace it with a better one from their neighborhood. (i.e.: Local Search [6], Tabu Search [7], Simulated Annealing [8]). Population-based methods: they iteratively evolve a population of solutions (i.e.: Genetic Algorithms [9,10], Particle Swarm [11]).

Metaheuristic optimization algorithms are often inspired from nature. According to the source of inspiration of the metaheuristic algorithms they can be classified into different categories. The main category is the biology-inspired algorithms which generally use biological evolution and/or collective behavior of animals. Science is another source of inspiration for the metaheuristics.

These algorithms are usually inspired physic and chemistry. Moreover, art-inspired algorithms [12] have been successful for the global optimization. They generally inspired from artists behavior to create artistic stuffs (such as musicians and architectures). Socially inspired algorithms can be defined as another source of inspiration and the algorithm simulate the social trying to extract the "swarm intelligence". In this paper, we use a social media inspired optimization algorithm: Twitter Optimization (TO) originally developed by Lv et al. on [13]. TO is able to solve combinatorial optimization problems by imitating human's social actions on Twitter: following, tweeting and retweeting.

As the most intelligent biological system, human society is considered very worthy of investigation. By observing the daily routine of humans, we surprisingly discovered that we ourselves, as individuals in human society, were actually performing some unconscious optimization tasks on the Internet, or more specifically on Twitter. This is because when we use Twitter, we tend to publish the Tweet (a term similar to Twitter) that is considered more valuable. For example, if someone receives two Tweets, separately on the local climate and the presidential elections, he would be more willing to share this last message on Twitter because it is worth mentioning. Similar filtering behavior occurs in all Twitter users. Everyone receives the Tweets filtered by others and publishes the Tweets filtered by himself.

Tweets that can be widely disseminated among users have been filtered millions of times and would be considered the most valuable Tweets at that time. This explains why Twitter users can always catch up with the access point. Inspired by this phenomenon, the algorithm called Twitter Optimization (TO) was created.

To create an effective algorithm for the phenomenon mentioned above, TO introduces three metaphors on Twitter. First, each Tweet is considered as a solution vector x composed of D real value parameters. And to minimize the optimization mission of the objective function F, the smaller the value $F(x)$, the more valuable the Tweet will be. Secondly, when a person publishes a Tweet, he will send the content to all his followers (a Twitter term represents a unidirectional relationship). This means that he produces a solution and shares it with the specific crowd connected to him.

Finally, each person must retweet (a Twitter term means to republish) the most valuable Tweet of the people who followed, implying that it is helping the best solution to spread more. The realization of the three metaphors makes TO look for the global optimum.

2 Social Media Optimization

The inspiration of this metaheuristic is taken from the interaction of people in social media, more specifically the behavior of people on Twitter. In order to define the behavior of the algorithm, it is imperative to detail the central elements that configure it. To do this, in the next section will delve into the formal definition of each of this elements.

Objective Function. Mathematical expression representing the problem to solve. In this case we apply the TO to solve the SCP.

Fitness. Represent the value of a solution or in this case a Tweet, evaluated in the objective function ($F(x)$). For example, if we are working with a minimization problem and evaluate two solution vectors: $F(x_1)$ and $F(x_2)$ with the objective function, you get the values 100 and 200 respectively, you could say that the vector x_1 is better, because it has a lower value than x_2.

Tweet. A tweet in case of this metaheuristic, corresponds to a feasible solution of the search space. A tweet represent vector of n columns. In binary case, possible values are 0 and 1 in each columns. Formally we define a new tweet as: $x_{new} = \{x_1, x_2, \ldots, x_n\}, \forall i \in \{1, 2, \ldots, n\}$ *and* $x_i \in \{0, 1\}$. Some metaheuristics work with binarization functions for the generation of solution vectors as shown in [14], however in TO, the formulas are presented in a general way but the decision variables (x_i), just can take the values $\{1, 0\}$.

Following. This behavior mimics human relations given in the context of tweeter. Initially the algorithm generates a direct graph between each of the people or tweeter users. Each user is allowed to follow only F users. Then, in each round, each user will update or optimize the following set through the operations of follow or unfollow. He will select the tweet with the best fitness received in this round and will follow his original author, if he does not follow it already. After this operation has been carried out correctly, the user who has the tweet with the worst fitness must be eliminated in such a way that F is preserved. The presented method makes the direct graph adjust as each round passes. This methodology ensures that p will be following the users that generate the best tweets, that is, the solutions closest to the global optimum.

Initialization of Algorithm. The process of tweeting involves generating a new solution and sharing it with the followers. During the initiation of the search algorithm, each user issues a tweet. The formal procedure is described in Eq. (1):

$$x_i^k = min_i + (max_i - min_i) * rand() \tag{1}$$

where x^k is the current solution of user k and x_i^k is the value of x^k in i-th dimension. min_i and max_i represent lower and upper value of solution value in i-th dimension.

The random retweet consists of the initialization process where a person k randomly selects a person p from $following_k$ and proceeds to retweet the p solution. Formally we can say $y_i^k = x_i^p$, where y^k is the solution that the user k will share and x^p is the current solution found by the user p.

Tweeting and Retweeting. The process of retweeting corresponds to the imitation of the process of sharing with followers, content that the user recognizes as valuable. This behavior ensures that a good solution is shared by as many users as possible. The TO algorithm adds two disturbance operators which are discussed below. When the user k decides to retweet the user's Tweet p, the user k randomly selects one of the following operations:

Comments:

$$x_{i\,new}^p = x_i^p + Character^k * rand(-1, 1) \tag{2}$$

Participation:

$$x_{i\,new}^p = x_i^k \tag{3}$$

When user decides to *comments* the Eq. (2) is calculated, for each dimension i, there is the possibility that the new solution is better than the current one, given that case, the new solution should replace the current one.

When a user decides to *participate*, he will contribute his own share to the activity. Given the above, Eq. (3) is used to simulate this behavior.

Publishing New Tweet. This specific operation consists of discarding a current solution to generate a new one. Additionally, users called *celebrity* are chosen in each round if they are ranked within the best 1% of all users and classified as *averageman* otherwise.

For the celebrity a new behavior is introduced which takes into consideration if its solution has not been updated during the last W times, given this case generates a new solution. Where W corresponds to the number of followers of k in the current round.

The behavior for *averagemen* in each round, will consist in generating a new Tweet according to the Eq. (4). The *celebrity* will choose the Tweet with better fitness and will own it.

$$x_i^k = x_{new_i} + \delta * rand(-1, 1) \tag{4}$$

where x_{new} is the best solution found so far, δ is the scan radius. The Eq. (4) is able to generate solutions near the best available solution. TO will stop when the number of iterations is greater than the upper limit N.

Hottest Tweet. The Hottest tweet or global optimum is defined as a solution $s^* \in S$ and it has a better fitness than all solutions of the search space, i.e., $\forall s \in S, f(s^*) \leq f(s)$.

A complete flow chart of the TO algorithm can be reviewed in the Fig. 1.

3 Set Covering Problem

SCP Formulation. The SCP is a well-known mathematical problem, which tries to cover a set of needs at the lowest possible cost. The SCP was included in the list of 21 $\mathcal{NP}$-*complet* problems of Karp [15]. There are many practical uses for this problem, such as: airline crew scheduling [16,17], location of emergency facilities [18], vehicle routing [19], network attack or defense [20], traffic assignment in satellite communication systems [21], the calculation of bounds in integer programs [22], assembly line balancing [23], political districting [24], among others. The SCP can be formulated as follows:

$$\text{Minimize} \quad Z = \sum_{j=1}^{n} c_j x_j \tag{5}$$

Subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \in I \tag{6}$$

$$x_j \in \{0, 1\} \quad \forall j \in J \tag{7}$$

Let $A = (a_{ij})$ be a $m \times n$ 0-1 matrix with $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ be the row and column sets respectively. We say that column j can be cover a row

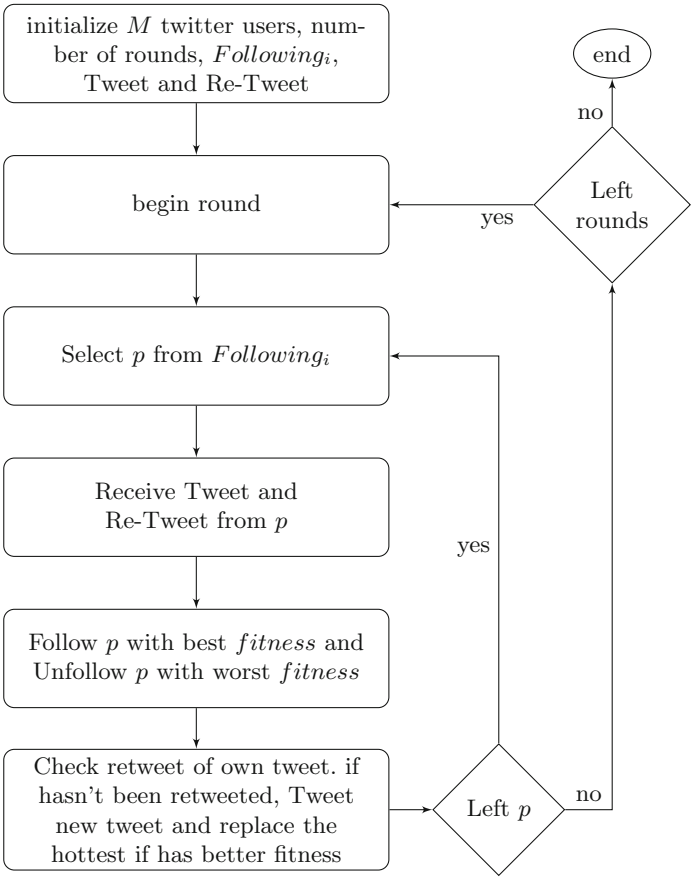

Fig. 1. Social media optimization workflow.

i if $a_{ij} = 1$. Where c_j is a nonnegative value that represents the cost of selecting the column j and x_j is a decision variable, it can be 1 if column j is selected or 0 otherwise. The objective is to find a minimum cost subset $S \subseteq J$, such that each row $i \in I$ is covered by at least one column $j \in S$.

4 Experiments

In order to test the correct execution of the T.O. metaheuristics, 30 independent executions were made of each instance of the benchmark. The metaheuristic T.O. measured its effectiveness against the instances of the library or benchmark OR (Beasley, 1990). T.O. was programmed in language Python 3.6, making use of the libraries Numpy and Scipy. GIT was used as source code repository. As a persistence, text files and in memory Data Base were used and as processing, a virtual machine with a 2.0 GHz (64 bit) dual-core CPU, 12 GB of RAM, 80 GB

Table 1. Solutions of SCP4.1 through T.O.

Instance	Optimum	Min	Max	Avg	RPD
SCP4.1	429	451	456	453.5	1.10
SCP4.2	512	611	675	643	19.34
SCP4.3	516	522	565	543.5	7.61
SCP4.4	494	514	548	531	6.20
SCP4.5	512	520	581	550.5	10.50
SCP4.6	560	566	648	607	12.65
SCP4.7	430	446	475	460.5	6.11
SCP4.8	492	499	526	512.5	5.13
SCP4.9	641	644	748	696	13.90
SCP4.10	514	543	580	561.5	6.38

of Hard Disk and Ubuntu 18.04 as operating system. In the table presented in this section, the RPD column reports the relative percentage deviation between the lowest experimentally obtained value (best fitness) and the global optimum for that instance (s *). The RPD is calculated by Eq. (8).

$$RPD = 100\frac{(Min - Optimum)}{Optimum} \tag{8}$$

As we can see in Table 1, the values of RPD are maintained between 1% and 19% for all evaluated instances. The instance with the best result was SCP4.1, obtaining an RPD of 1.10%, with a minimum value of 461. The worst result was obtained in the instance SCP4.2, where the value of the RPD was 19.34%. The regular behavior of the algorithm without changes in its parameters was around 6%. In terms of RPD evaluation, the algorithm behaved consistently in all evaluated instances.

If we look at Fig. 2, a fast convergence of the algorithm towards the global optimum is verified, achieving an asymptotic behavior to y axis, managing to escape from local optima successfully. The behavior of the algorithm before the benchmark has demonstrated its functionality and effectiveness.

Additionally, a comparison of the results obtained when solving the SCP was made, with two known algorithms. Specifically, TO is compared with Harmony Search (HS) in the work done by Salas et al. in [25] and with Black Hole in the work done in [26]. As can be seen in Table 2, the TO algorithm was superior in 9/10 instances, achieving an outstanding result.

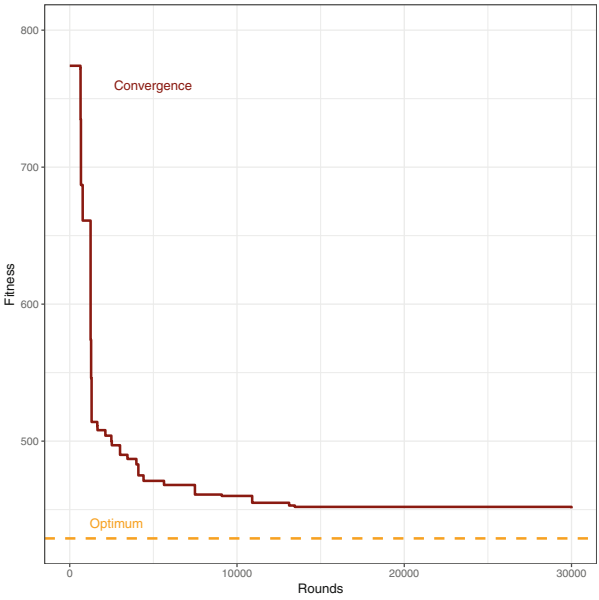

Fig. 2. Convergence to optimum, instance SCP41.

Table 2. Comparison of algorithms for SCP resolution

Instance	Optimum	TO	HS	BH	Best algorithm
SCP4.1	429	451	468	455	**TO**
SCP4.2	512	611	611	544	BH
SCP4.3	516	522	587	551	**TO**
SCP4.4	494	514	569	527	**TO**
SCP4.5	512	520	581	548	**TO**
SCP4.6	560	566	648	601	**TO**
SCP4.7	430	446	495	461	**TO**
SCP4.8	492	499	560	528	**TO**
SCP4.9	641	644	775	688	**TO**
SCP4.10	514	513	596	547	**TO**

5 Conclusion

Basically the social capacity to relate between people, allows us to be able to imitate their behavior to obtain relevant topics in their interaction. As indicated in the introduction to this paper, people are more willing to share information they consider valuable or important, rather than of little value. This very natural behavior allows to be applied in algorithms that behave in a good way in the resolution of problems like the SCP.

Our experiment results show good results solving SCP benchmark functions. Some improvements can be implemented taking ideas from other metaheuristic techniques, that introduce certain operators that better aspects such as exploitation and exploration. This is considered as future work. Finally It can be concluded that TO has a potential in solving optimization problems.

References

1. Talbi, E.: Metaheuristics: From Design to Implementation, vol. 74. Wiley, Hoboken (2009)
2. Beheshti, Z., Shamsuddin, S.: A review of population-based meta-heuristic algorithms. Int. J. Adv. Soft Comput. Appl. **5**(1), 1–35 (2013)
3. Festa, P., Resende, M.: GRASP: an annotated bibliography. In: Essays and Surveys in Metaheuristics. ORCS, vol. 15, pp. 325–367. Springer, Boston (2002). https://doi.org/10.1007/978-1-4615-1507-4_15
4. Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.): Ant Colony Optimization and Swarm Intelligence. LNCS, vol. 5217. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87527-7
5. Shah, H.: The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Comput. **1**(1–2), 71–79 (2009)
6. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **28**(3), 392–403 (1998)
7. Nowicki, E., Smutnicki, C.: A fast tabu search algorithm for the permutation flowshop problem. Eur. J. Oper. Res. **91**(1), 160–175 (1996)
8. van Laarhoven, P.J.M., Aarts, E.H.L.: Simulated annealing. In: Simulated Annealing: Theory and Applications. MAIA, vol. 37, pp. 7–15, Springer, Dordrecht (1987). https://doi.org/10.1007/978-94-015-7744-1_2
9. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)
10. Glover, F., Laguna, M., Martí, R.: Scatter search and path relinking: advances and applications. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics. ISOR, vol. 57, pp. 1–35. Springer, Boston (2003). https://doi.org/10.1007/0-306-48056-5_1
11. Kennedy, J.: Particle swarm optimization. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 760–766. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-30164-8
12. Patnaik, S., Yang, X.-S., Nakamatsu, K.: Nature-Inspired Computing and Optimization, vol. 10. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-50920-4
13. Lv, Z., Shen, F., Zhao, J., Zhu, T.: A swarm intelligence algorithm inspired by Twitter. In: Hirose, A., Ozawa, S., Doya, K., Ikeda, K., Lee, M., Liu, D. (eds.) ICONIP 2016. LNCS, vol. 9949, pp. 344–351. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46675-0_38
14. Crawford, B., Soto, R., Astorga, G., García, J., Castro, C., Paredes, F.: Putting continuous metaheuristics to work in binary search spaces. Complexity **2017**, 8404231:1–8404231:19 (2017)
15. Karp, R.: Reducibility among combinatorial problems. In: Jünger, M., et al. (eds.) 50 Years of Integer Programming 1958–2008 - From the Early Years to the State-of-the-Art, pp. 219–241. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-540-68279-0_8

16. Marchiori, E., Steenbeek, A.: An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: Cagnoni, S. (ed.) EvoWorkshops 2000. LNCS, vol. 1803, pp. 370–384. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45561-2_36
17. Housos, E., Elmroth, T.: Automatic optimization of subproblems in scheduling airline crews. Interfaces **27**(5), 68–77 (1997)
18. Farahani, R.Z., Asgari, N., Heidari, N., Hosseininia, M., Goh, M.: Covering problems in facility location: a review. Comput. Ind. Eng. **62**(1), 368–407 (2012)
19. Daskin, M.S., Stern, E.H.: A hierarchical objective set covering model for emergency medical service vehicle deployment. Transp. Sci. **15**(2), 137–152 (1981)
20. Intanagonwiwat, C., Estrin, D., Govindan, R., Heidemann, J.: Impact of network density on data aggregation in wireless sensor networks. In: Proceedings 22nd International Conference on Distributed Computing Systems, pp. 457–458, July 2002. https://doi.org/10.1109/ICDCS.2002.1022289. ISSN: 1063-6927
21. Ribeiro, C.C., Minoux, M., Penna, M.C.: An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment. Eur. J. Oper. Res. **41**(2), 232–239 (1989)
22. Fisher, M.L.: The Lagrangian relaxation method for solving integer programming problems. Manag. Sci. **27**(1), 1–18 (1981)
23. Beasley, J.E.: An algorithm for set covering problem. Eur. J. Oper. Res. **31**(1), 85–93 (1987)
24. Garfinkel, R.S., Nemhauser, G.L.: The set-partitioning problem: set covering with equality constraints. Oper. Res. **17**(5), 848–856 (1969)
25. Salas, J., Mora, M., Barriga, H., Rubio, J., Broderick, C.: Study of population variation using harmony search for the resolution of set covering problem. Revista Científica de la UCSA **4**(3), 20–33 (2017)
26. Rubio, Á.G., et al.: An binary black hole algorithm to solve set covering problem. In: Fujita, H., Ali, M., Selamat, A., Sasaki, J., Kurematsu, M. (eds.) IEA/AIE 2016. LNCS, vol. 9799, pp. 873–883. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42007-3_74