Journal of Software Engineering and Applications, 2022, 15, 262-273
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Generic Tabu Search

Chadi Kallab[1], Samir Haddad[1]*, Imad El-Zakhem[1], Jinane Sayah[2], Mohamad Chakroun[3], Nisrine Turkey[4], Jinan Charafeddine[5], Hani Hamdan[6], Wafaa Shakir[7]

[1]Department of Computer Science and Mathematics, Faculty of Arts and Sciences, University of Balamand, Koura, Lebanon
[2]Department of Telecom and Networks, Issam Fares Faculty of Technology, University of Balamand, Koura, Lebanon
[3]Faculty of Computer Science and Electrical Engineering, Universität Rostock, Rostock, Germany
[4]Faculty of Engineering, Notre Dame University, Jounieh, Lebanon
[5]Université Paris-Saclay, Pôle scientifique et technologique de Vélizy, Laboratoire d'Ingénierie des Systèmes de Versailles (LISV EA4048), Vélizy, France
[6]Université Paris-Saclay, CentraleSupélec, CNRS, Laboratoire des Signaux et Systèmes (L2S UMR CNRS 8506), Gif-sur-Yvette, France
[7]Department of Computer Systems, Faculty of Engineering, Al-Furat Al-Awsat Technical University, Babil, Iraq
Email: Chadi.Kallab@fty.balamand.edu.lb, *Samir.Haddad@balamand.edu.lb, IZhakhem@balamand.edu.lb, Jinane.Sayah@balamand.edu.lb, Mohamad.Chakroun@gmail.com, NTurkey@ndu.edu.lb, Jinan.Charafeddine@lisv.uvsq.fr, Hani.Hamdan@centralesupelec.fr, inb.Wfa@atu.edu.iq

How to cite this paper: Kallab, C., Haddad, S., El-Zakhem, I., Sayah, J., Chakroun, M., Turkey, N., Charafeddine, J., Hamdan, H. and Shakir, W. (2022) Generic Tabu Search. *Journal of Software Engineering and Applications*, **15**, 262-273.
https://doi.org/10.4236/jsea.2022.157016

Received: June 15, 2021
Accepted: July 25, 2022
Published: July 28, 2022

Abstract

The Multiple Sequence Alignment problem is considered to be an NP-Hard problem, requiring initially a specific encoding schema and design, as for any other of its siblings, to implement and run any of the main categories of heuristic. This paper intends to discuss our proposed generic implementation of the Tabu Search algorithm, a heuristic procedure proposed by Fred Glover to solve discrete combinatorial optimization problems. In this research, we try to coordinate and synchronize different designs/implementations discussed in many literatures, with some of the references mentioned in this paper. The basic idea is to avoid that the search for best solutions stops when a local optimum is found, by maintaining a list of non-acceptable or forbidden (taboo) solutions/costs, called Tabu list or Short-Term Memory (STM). In our algorithm, we attempt to add some executions tracing functionalities in order to help later analysis for initial parameters tuning. On the other hand, we propose to include the concept of a list called Long-Term Memory (LTM), so that some of the best solutions found so far can be saved, for search diversification.

Keywords

Generic, Heuristics, Bio-Informatics, NP-Hard, Tabu Search, STM, LTM

1. Introduction

One of the many problems that are considered to be NP-Hard is the Multiple

Sequence Alignment one that initially requires, as for any other of its siblings, a specific encoding schema and design of the main functionalities of the heuristics algorithm being implemented and executed. This design was supported by references [1] [2] [3] [4].

The main issue with that problem, as discussed in our ICeND2013 conference [1], was to come up with an encoding that would be suitable for the different Heuristics algorithms inspired by papers [5] [6] [7] [8]. Once the encoding is agreed on, the remaining part of designing and building the algorithm that will get a solution as close to the optimum as possible would be nonetheless as important. We have decided in our research to focus on the Tabu Search heuristic. However, we discovered that to be able to adjust some of the flaws of the standard algorithm, we improved it by adding functionalities like tracing the executions, so that later analysis could help tune the initial parameters better.

In this paper, we'll be discussing the implementation of an advanced version of the Tabu Search algorithm, adding to the standard version some modifications such as the tracing functionalities, mainly through the use of different kinds of memories, short-term (STM) and long-term (LTM) instead of one, for solutions checked as Tabu or as good fit, along with some useful parameters.

The problem in the standard algorithm would be that, despite "unfit" solutions would be moved/marked as Tabu, so they won't be handled in further iteration, there is no guarantee that the algorithm would get out of a local optimum which could eventually be far from the effective best solution.

2. Initial Algorithm

Standard Algorithm

Tabu Search (TS) is a heuristic procedure proposed by Fred Glover to solve discrete combinatorial optimization problems. The basic idea is to avoid that the search for best solutions stops when a local optimum is found, by maintaining a list of non-acceptable or forbidden (taboo) solutions/costs, called Tabu list or Short-Term Memory (STM). We have found many papers discussing different implementations of TS, amongst them references [9] [10] [11] [12] [13]. Other papers in literature suggest some hybrid or generic implementation of the initial heuristic procedure, as in reference [14]-[19], while keeping focus on the problem being tackled.

Advanced TS algorithms suggest that some of the best solutions found so far be saved for search diversification in a list called Long-Term Memory (LTM).

The use of these memory lists bring to light the fact that the updating process of the current and best solution does ignore those that were marked in the lists, which grow and shrink per iteration. Occasionally, moving the current solution to a "forbidden" one is allowed given a certain "aspiration" criteria, usually involving an improvement in cost from the current one.

As opposed to other algorithms, the current solution of the inner loop next iteration is selected from a set of N neighbor solutions, deduced from the actual current one by perturbing it N times. This solution will be overwritten if, at the beginning of the next iteration, a better solution was previously acknowledged in memory.

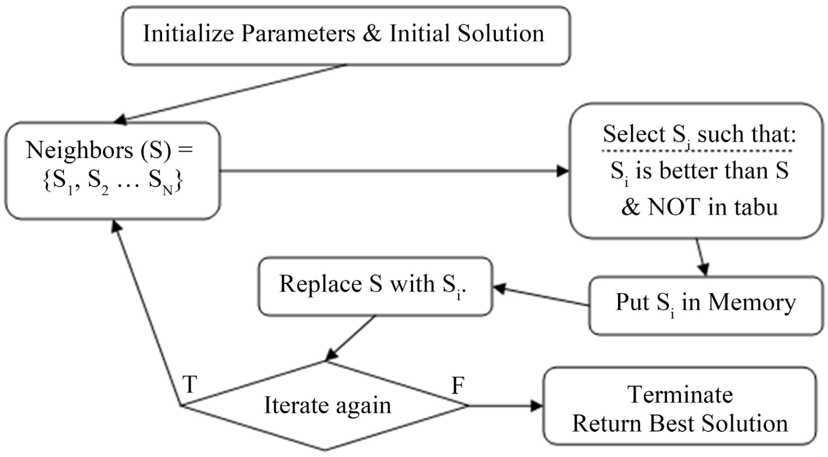


Figure 1. Standard Tabu search general flow diagram.

As shown in **Figure 1**, the standard algorithm starts from an initial solution and set of parameters then repeatedly, until a certain stopping condition is satisfied, creates a candidate list of solutions called neighbors, evaluates and chooses the best admissible one among them, updates the STM and acquisition components with the selection, then replace the current solution with the one newly selected.

3. Proposed Algorithm

The main idea behind Tabu Search algorithm was to simulate placing solution as Tabu, when their evaluation will be more costly to the algorithm in further iterations, thus avoiding using them as potential neighbor. However, most literature about the standard algorithm don't mention any approach to avoid getting stuck in a local optimum area, since that best solution is not going to be moved to Tabu. Because of this gap in literature, we decided in our algorithm to give the opportunity to remember a previously best fit solution as a potential optimum instead of the best one, into a long-term memory. Adding more memories into the algorithm came supported by the procedure that the human brain follows to shift selected information from its STM (also known as daily memory) into its LTM (mostly storing/recalling experiences from the past).

The steps of our proposed algorithm are almost very similar to those of the basic standard algorithm, with the difference that some of them, highlighted in blue, offer the possibility to later on trace back each execution and allow better analysis, thus resulting in eventual update of initial parameters. The words/fragments highlighted in green are more of structure and/or behavior enhancements that try to bridge the gap between the initial annealing process and our simulated annealing algorithm.

3.1. Main Procedure

Inputs: Initial Solution S_0, Number of Iterations *numIterations*,
Number of Neighbors *numNeighbors*,

Precondition: $numIterations > 0$ $numNeightbors > 0$

```
Outputs:    Optimal Solution BestS,
Algorithm:
    Assign to "CurS" the value of the initial solution "S0"
    Compute estimate of "CurS" into "CurEstimate"
    Assign "CurS" and estimate to {"BestS", "BestEstimate"}
    Create a dynamic list "LTM" of solutions and estimates
    Initialize index "i" to 1
    While True do
        If $stopLoop()$ or $i > numIterations$ then
            Exit While
        Update LTM with current and/or best solutions
        Reset the STM memory
        {{Current Solution and Estimate are selected}}
        {{        from LTM first if possible}}
        Select best fit between LTM and STM into "CurS"
        Compute estimate of "CurS" into ( $CurEstimate$ )
        Save in history "CurrS"    {{history ≠ memory}}
        Add to STM values of "CurS" and "CurEstimate"
        Generate a list of "Neighbors" from " $CurS$ "
        For "j" from $1$ to "numNeighbors" do
            Assign to "Sol" neighbor solution at "j"
            Compute estimate of "Sol" into "SolEstimate"
            {{If neighbor is NOT tabu OR tabu}}
            {{   but aspires to be better than CurS}}
            {{Otherwise, consider the neighbor as Tabu}}
            If $NOT\{isTabu(Sol)\}$ Or $aspiration(Sol, CurS)$   Then
                Assign "Sol" and "SolEstimate" to current
                {{Update best solution so far}}
                {{   if neighbor aspires to be the best}}
                If $aspiration(Sol, CurS)$ Then
                    Assign "Sol" and "SolEstimate" to best
            Else-If $NOT\{isTabu(Sol)\}$ And $NOT\{aspiration(Sol, CurS)\}$
            Then
                Add to STM value of " $SolEstimate$ "
        Save in history {"BestS", "BestEtimate", "CurS", "CurEstimate"
            "Neighbors", "LTM", "STM"}
        Increment the index "i" by 1
```

The proposed algorithm, discussed in the main steps mentioned above and in Figure 2, differs from the standard one mainly by the introduction two update steps before and after checking the stopping/iterating condition: first applies on the two running (current and best) solutions, while the other one modifying the different memory components is applied initially and after the condition check. Note that after updating those components, the algorithm will be able to switch

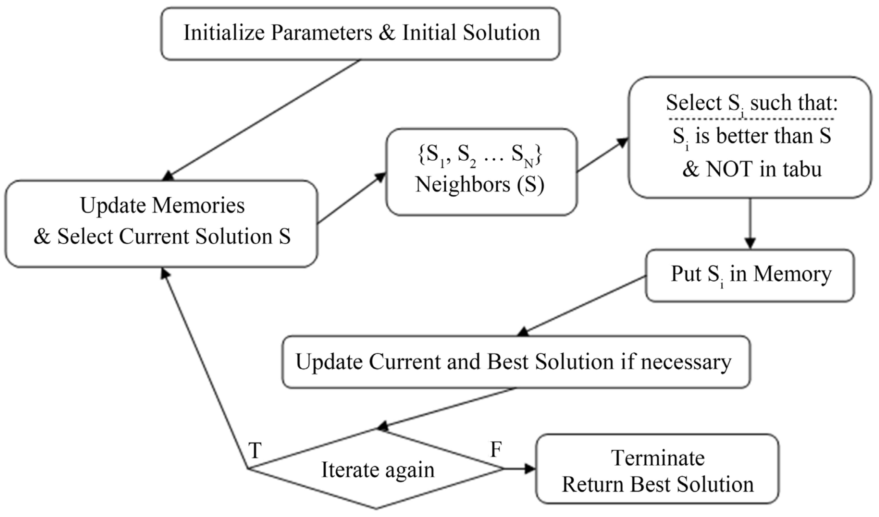


Figure 2. Advanced Tabu search general flow diagram.

the current solution with a best fit previously evaluated and selected as a potential optimal solution.

3.2. Object-Oriented Design

In order to implement the algorithm steps in **Figure 2**, we designed our object-oriented components in a simple way as shown below in **Figure 3** with suggested classes representing some problems in **Figure 4**, since the flexibility we are offering is much more in the steps themselves. In order to simplify the suggested algorithm, we considered the memory components as lists of solutions, and the selection and/or memorization of solutions as procedures to be handled directly by the algorithm's main steps.

The main component "TSProblem", on the upper-right, is intended to hold properties managing the different parameters, components and methods the algorithm will need while executing its steps, whether generic and/or purpose-specific, as for ex: the ability to create a new solution or get a new neighbor.

The class "TSSolution", just at its left, is intended to hold properties managing the different values encapsulated in each solution handled during execution, mainly those related to the evaluation process.

Last but not least, the "TabuSearch" base algorithm with its two base/super classes, show the main components being handled and functionalities being executed per iteration. The reason behind splitting them into three classes was to set the ground for potential use of some methodologies with other algorithms that might find some common grounds, and benefit from Object-Oriented development as much as possible.

In order to prove that our implementation is generic enough, we designed some basic problems, where we can have results to compare with those of the algorithm, in a simple way. For simplicity and efficiency, we chose to work with simple math formula "Formula" fed with values of radix {2, 8, 10 or 16}. We also have implemented each radix-related problem as both minimization and maximization.

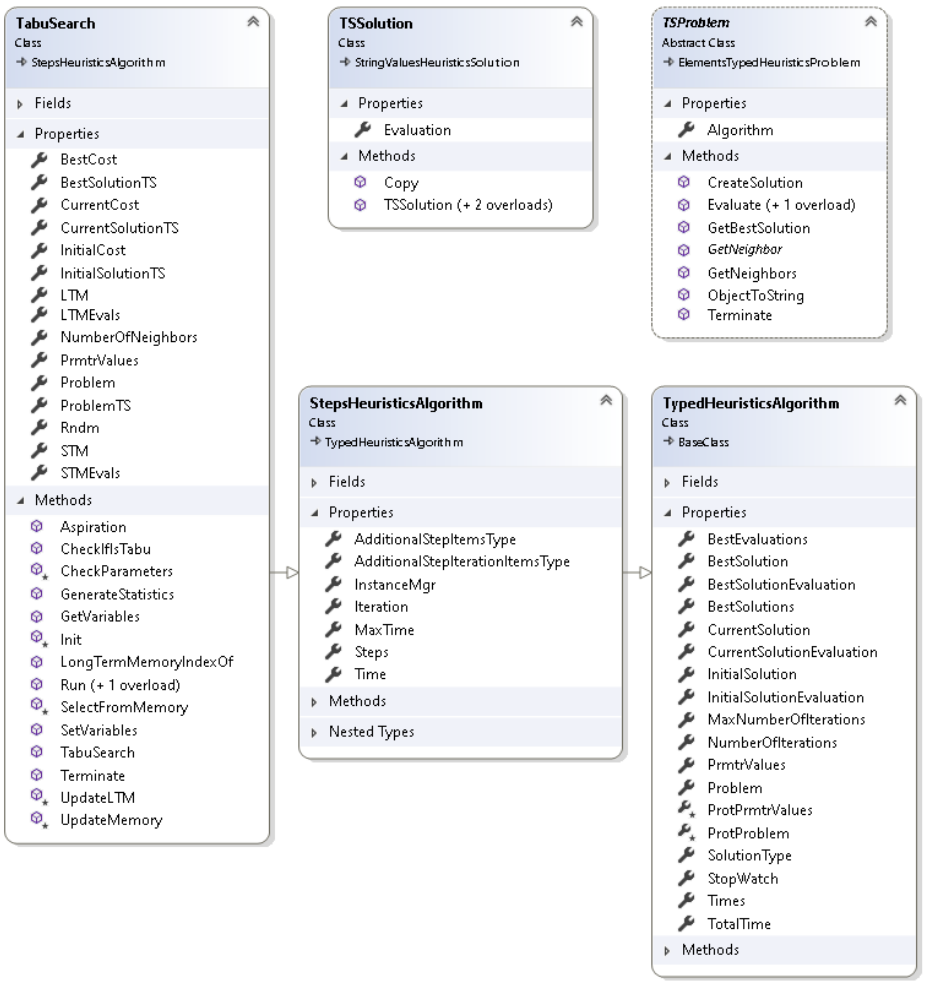


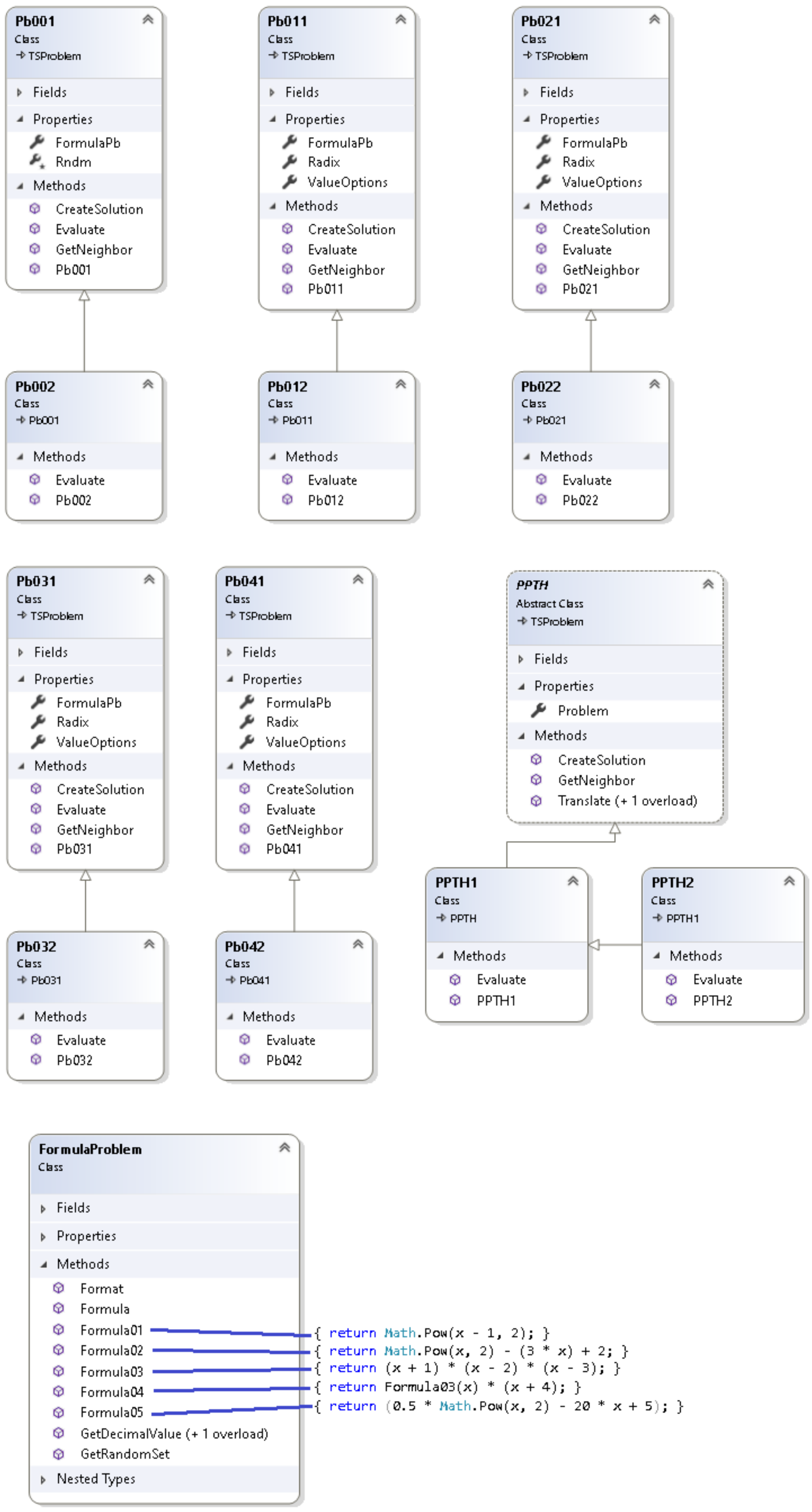
Figure 3. Suggested TS components class diagram.

In this implementation, only the estimates are moved to the STM list, instead of the whole solution's properties. Therefore, evaluating if a solution is tabu becomes as simple as evaluating its penalty. The aspiration method implemented relies on finding the minimal or maximal solution estimate.

If the Tabu Search algorithm deals with minimization, the estimate of a solution should be the result of the evaluation function, mentioned in a previous chapter, and multiplied by −1 otherwise. For both approaches, the result of the neighbor function for a given solution is, in other words, a set of results of the perturbation of this solution.

The fact that TS may deal, with either minimization or maximization, yields the obligation to implement, consequently, a method used during the aspiration, and that returns the best solution between 2 solutions fed as parameters.

In order to illustrate the generic implementations applied on these problems, we have developed a quick straight forward windows application, in which we dynamically modified the parameters a default algorithm based on the suggested generic implementation, instead of writing many versions inheriting for our main implementation.



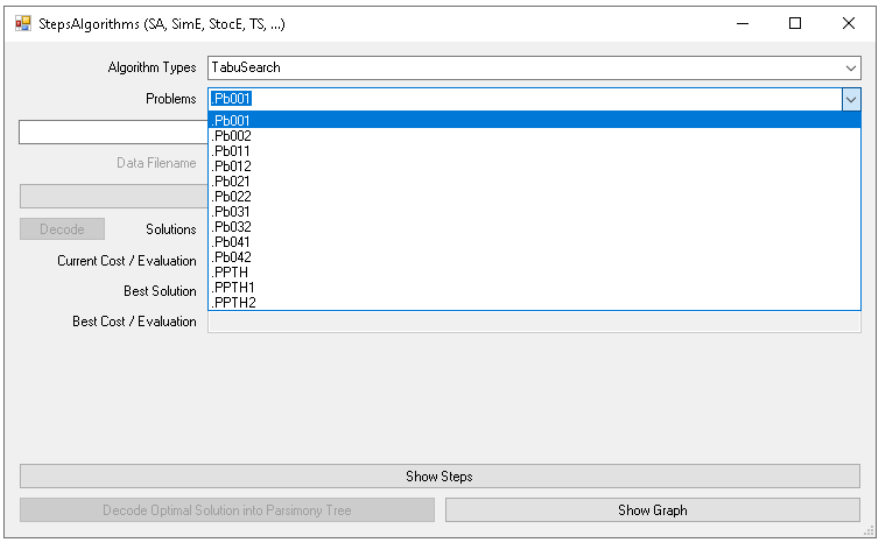
```
{ return Math.Pow(x - 1, 2); }
{ return Math.Pow(x, 2) - (3 * x) + 2; }
{ return (x + 1) * (x - 2) * (x - 3); }
{ return Formula03(x) * (x + 4); }
{ return (0.5 * Math.Pow(x, 2) - 20 * x + 5); }
```

Figure 4. Some supporting problem implementations.

Figure 5 shows the first step in running the algorithm, which is the problem selection. In our windows application, we populated this drop-down dynamically from a folder containing problems included in our package.

Next step, shown in Figure 6 and Figure 7, is about parametrization of the algorithm run, whether concerning properties of the problem (environment) itself or the execution's methods. Even though default values were given to those parameters, we advise to modify them to be able to view effective results, and/or test different cooling schedules.

The third and final step, as shown in Figure 8, is to simply run the algorithm. In the above screenshot, we are showing the generated result of an execution on problem Pb001, along with the actual value corresponding to the last current solution compared to the cost evaluated during the last iteration of the algorithm.

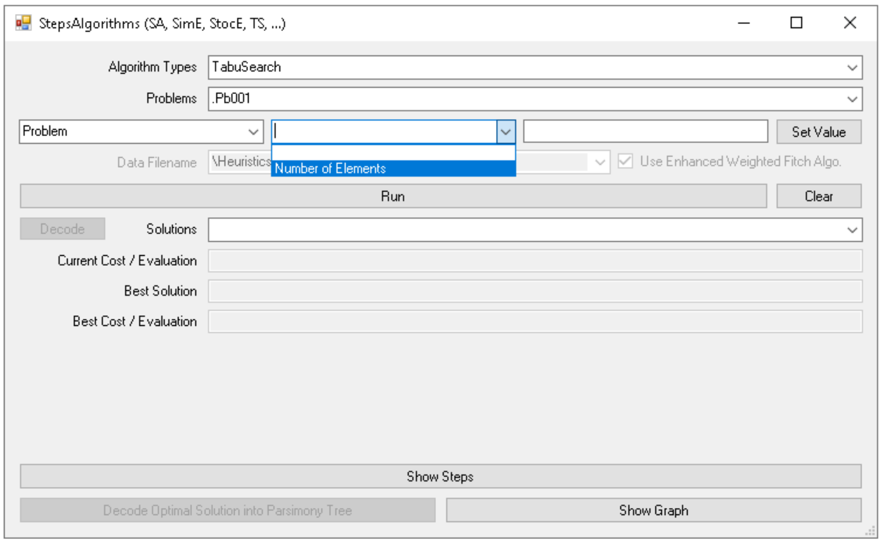

StepsAlgorithms (SA, SimE, StocE, TS, ...)

Algorithm Types: TabuSearch

Problems: .Pb001

.Pb001
.Pb002
.Pb011
.Pb012
.Pb021
.Pb022
.Pb031
.Pb032
.Pb041
.Pb042
.PPTH
.PPTH1
.PPTH2

Figure 5. Win App, TS execution—problem selection.


StepsAlgorithms (SA, SimE, StocE, TS, ...)

Algorithm Types: TabuSearch

Problems: .Pb001

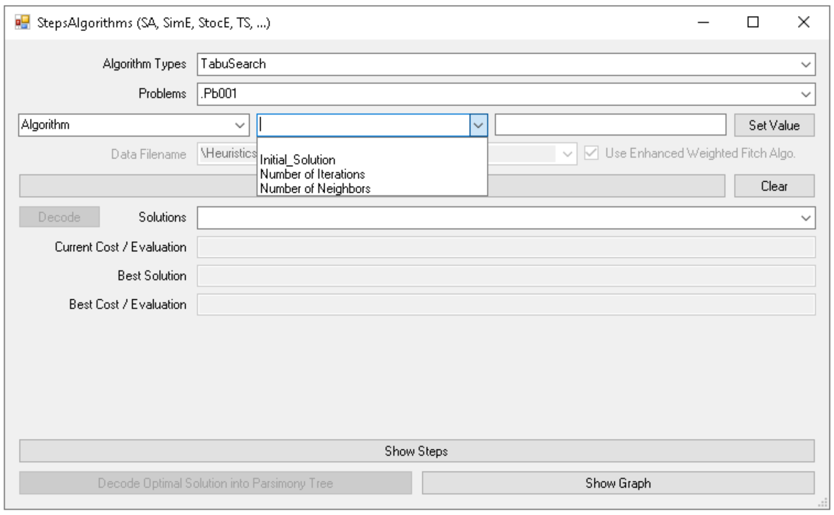
Problem

Number of Elements

Data Filename: \Heuristics

Use Enhanced Weighted Fitch Algo.

Figure 6. Win App, TS exec.—problem parameterization.


StepsAlgorithms (SA, SimE, StocE, TS, ...)

Algorithm Types: TabuSearch

Problems: .Pb001

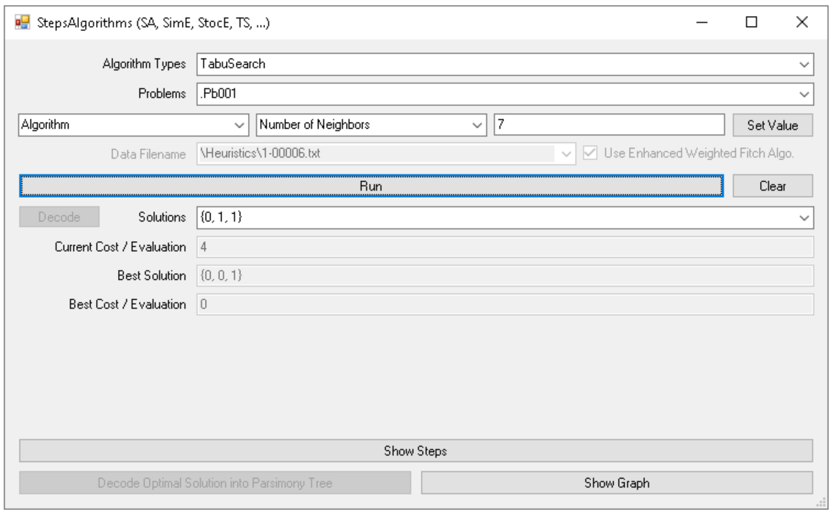
Algorithm

Initial_Solution
Number of Iterations
Number of Neighbors

Data Filename: \Heuristics

Set Value
Use Enhanced Weighted Fitch Algo.
Clear
Decode
Solutions
Current Cost / Evaluation
Best Solution
Best Cost / Evaluation
Show Steps
Decode Optimal Solution into Parsimony Tree
Show Graph

Figure 7. Win App, TS exec.—algorithm parameterization.


StepsAlgorithms (SA, SimE, StocE, TS, ...)

Algorithm Types: TabuSearch

Problems: .Pb001

Algorithm: Number of Neighbors: 7

Data Filename: \Heuristics\1-00006.txt

Solutions: {0, 1, 1}

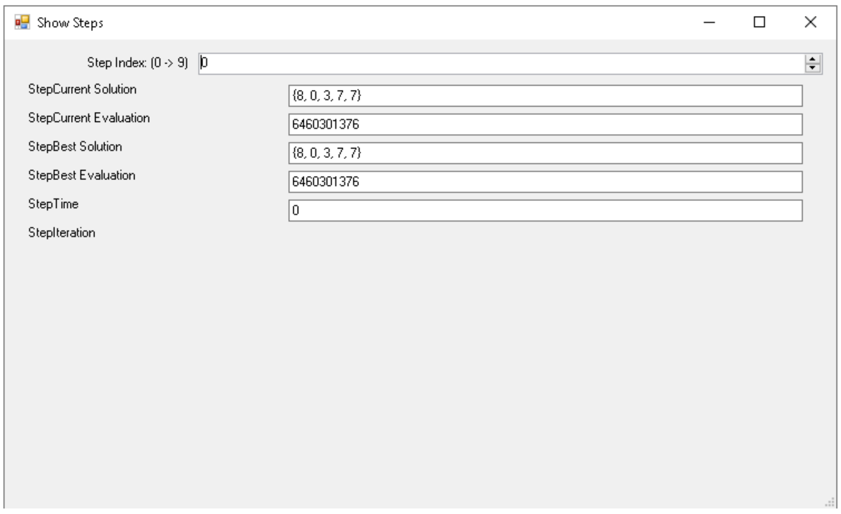
Current Cost / Evaluation: 4

Best Solution: {0, 0, 1}

Best Cost / Evaluation: 0

Set Value
Use Enhanced Weighted Fitch Algo.
Run
Clear
Decode
Show Steps
Decode Optimal Solution into Parsimony Tree
Show Graph

Figure 8. Win App, TS execution—sample run.


Show Steps

Step Index: (0 -> 9): 0

StepCurrent Solution: {8, 0, 3, 7, 7}

StepCurrent Evaluation: 6460301376

StepBest Solution: {8, 0, 3, 7, 7}

StepBest Evaluation: 6460301376

StepTime: 0

StepIteration

Figure 9. TS Tracing functionality—initial iteration.

The next section of the screen shows the optimal solution it was able to detect with a cost of 0, which is mathematically the actual best cost. For more complex problems, running the algorithm for more time will most probably change the best cost value converging towards a more suitable optimum.

Tracing back the execution of the execution mentioned we can see in **Figure 9** that the algorithm reached 80377 as the best solution at the end of the initial step.

4. Conclusion

This research can be developed by trying to design and implement solutions to the below issues raised during our research:

➢ Fixed Input Parameters:

○ "Tabu Search" explores the search space for a given number of iterations, and a number of neighbors per iteration.

➢ Applying benchmarks for more accurate analysis and validation of the alternatives

○ Use benchmarks for different problem encodings have to be applied.

➢ Randomness doesn't handle potentially repetitions:

○ In Neighbor method

➢ Initial solution may be far from the optimal one, thus the algorithm will take more time.

➢ Moving a solution to and from Tabu lists is not dynamically efficient, as it doesn't really account or take into consideration the potential effect of the values of some fields or variables induced by previous moves.

➢ Inspired by references [10] [11] [14] and [16] amongst others, more efforts can be put into making more use of the LTM and possible introduction of other types of memory.

➢ Another future work idea would be work on the memory components by making them more self-manageable through encapsulating the updating and selection procedure, then enriching the system with other types of memory components that would allow potential intensification and diversification during the algorithm run. This idea is further supported by references [20] to [28].

➢ A second future work could be to allow the main procedure of generating neighbors and selecting from memories "smart" enough to gain time and search further more.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Kallab, C. (2013) Generic Encoding and Phylogenies. *Proceedings of the ICeND Con-*

ference, Kuala Lumpur, 4-6 March 2013, 142.

[2] Kim, J. and Warnow, T. (1999) Tutorial on Phylogenetic Tree Estimation. https://scholar.google.com/scholar?q=Kim%20J.,%20Warnow%20T.%20Tutorial%20on%20phylogenetic%20tree%20estimation,%201999

[3] Moret, B., Bader, D. and Warnow, T. (2002) High-Performance Algorithm Engineering for Computational Phylogenetics. *The Journal of Supercomputing*, **22**, 99-111. https://link.springer.com/article/10.1023/A:1014362705613 https://doi.org/10.1023/A:1014362705613

[4] Opper, D. (2005) Parsimony Phylogenetic Trees. http://www.icp.ucl.ac.be/~opperd/private/parsimony.html

[5] Stamatakis, A., Ott, M. and Ludwig, T. (2005) RAxML-OMP: An Efficient Program for Phylogenetic Inference on SMPs. In: Malyshkin, V., Ed., *Parallel Computing Technologies. PaCT* 2005, Lecture Notes in Computer Science, Vol. 3606, Springer, Berlin, 288-302. https://doi.org/10.1007/11535294_25

[6] Felsenstein, J. (1982) Numerical Methods for Inferring Evolutionary Trees. *The Quarterly Review of Biology*, **57**, 379-404. https://doi.org/10.1086/412935

[7] Fitch, W. (1971) Toward Defining the Course of Evolution: Minimum Change for a Specified Tree Topology. *Systematic Zoology*, **20**, 406-416. https://doi.org/10.2307/2412116

[8] Hendy, M.D. and Penny, D. (1982) Branch and Bound Algorithms to Determine Minimal Evolutionary Trees. *Mathematical Biosciences*, **59**, 277-290. https://www.sciencedirect.com/science/article/abs/pii/002555648290027X https://doi.org/10.1016/0025-5564(82)90027-X

[9] Battiti, R. and Tecchiolli, G. (1994) The Reactive Tabu Search. *ORSA Journal on Computing*, **6**, 126-140. https://doi.org/10.1287/ijoc.6.2.126

[10] Burke, E., De Causmaecker, P. and Vanden Berghe, G. (1999) A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem. 2*nd Asia-Pacific Conference on Simulated Evolution and Learning*, Vol. 1585, 187-194. https://doi.org/10.1007/3-540-48873-1_25

[11] Crainic, T.G. and Gendreau, M. (1999) Towards an Evolutionary Method—Cooperative Multi-Thread Parallel Tabu Search Heuristic Hybrid. In: Voss, S., Martello, S., Osman, I.H. and Roucairol, C., eds., *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Dordrecht, 331-344. https://doi.org/10.1007/978-1-4615-5775-3_23

[12] Crainic, T.G., Gendreau, M. and Farvolden, J.M. (2000) A Simplex-Based Tabu Search for the Multicommodity Capacitated Fixed Charge Network Design Problem. *INFORMS Journal on Computing*, **12**, 223-236. https://doi.org/10.1287/ijoc.12.3.223.12638

[13] Crainic, T.G., Gendreau, M., Soriano, P. and Toulouse, M. (1993) A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements. *Annals of Operations Research*, **41**, 359-383. https://doi.org/10.1007/BF02023001

[14] Gendreau, M. (2002) Recent Advances in Tabu Search. In: Ribeiro, C.C. and Hansen, P., Eds., *Essays and Surveys in Metaheuristics*, Kluwer, Dordrecht, 369-377. https://doi.org/10.1007/978-1-4615-1507-4_16

[15] Basu, S. (2012) Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. *American Journal of Operations Research*, **2**, Article No. 19930. https://doi.org/10.4236/ajor.2012.22019

[16] Schweiger, K. and Sahamie, R. (2013) A Hybrid Tabu Search Approach for the De-

sign of a Paper Recycling Network. *Transportation Research Part E: Logistics and Transportation Review*, **50**, 98-119. https://doi.org/10.1016/j.tre.2012.10.006

[17] Sujitjorn, S., Kluabwang, J., Puangdownreong, D. and Sarasiri, N. (2009) Adaptive Tabu Search and Management Agent. *The ECTI Transactions on Electrical Engineering, Electronics, and Communications* (*ECTI-EEC*), **8**, 1-10.

[18] Devarenne, I., Mabed, H. and Caminada, A. (2006) Intelligent Neighborhood Exploration in Local Search Heuristics. 18*th IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI*'06), Washington DC, 13-15 November 2006, 144-150. https://doi.org/10.1109/ICTAI.2006.68

[19] Schaerf, A. (1996) Tabu Search Techniques for Large High-School Timetabling Problems. *IEEE Transactions on Systems Man and Cybernetics—Part A Systems and Humans*, **29**, 368-377. https://doi.org/10.1109/3468.769755

[20] Fink, A. and Voss, S. (2002) Generic Application of Tabu Search Methods to Manufacturing Problems. *SMC*'98 *Conference Proceedings*. 1998 *IEEE International Conference on Systems, Man, and Cybernetics* (*Cat. No.*98*CH*36218), San Diego, 14 October 1998, 2385-2390.

[21] Glover, F. and Laguna, M. (1997) Tabu Search (Vol. 22). Kluwer Academic Publishers, Boston. https://doi.org/10.1007/978-1-4615-6089-0

[22] Santos, H.G., Ochi, L.S. and Souza, M.J. (2005) A Tabu Search Heuristic with Efficient Diversification Strategies for the Class/Teacher Timetabling Problem. *Journal of Experimental Algorithms* (*JEA*), **10**, 2-9. https://doi.org/10.1145/1064546.1180621

[23] Glover, F., Lü, Z.P. and Hao, J.-K. (2010) Diversification-Driven Tabu Search for Unconstrained Binary Quadratic Problems. Springer, Berlin. https://doi.org/10.1007/s10288-009-0115-y

[24] James, T., Rego, C. and Glover, F. (2009) Multistart Tabu Search and Diversification Strategies for the Quadratic Assignment Problem. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, **39**, 579-596. https://doi.org/10.1109/TSMCA.2009.2014556

[25] Duarte, A. and Martí, R. (2007) Tabu Search and GRASP for the Maximum Diversity Problem. *European Journal of Operational Research*, **178**, 71-84. https://doi.org/10.1016/j.ejor.2006.01.021

[26] Laporte, G., Potvin, J.-Y. and Quilleret, F. (1997) A Tabu Search Heuristic Using Genetic Diversification for the Clustered Traveling Salesman Problem. *Journal of Heuristics*, **2**, 187-200. https://doi.org/10.1007/BF00127356

[27] Santos, H.G., Ochi, L.S. and Souza, M.J.F. (2005) A Tabu Search Heuristic with Efficient Diversification Strategies for the Class/Teacher Timetabling Problem. *ACM Journal of Experimental Algorithms*, **10**, 2.9. https://doi.org/10.1145/1064546.1180621

[28] Liu, G.-Y., He, Y., Fang, Y.H. and Qiu, Y.H. (2004) A Novel Adaptive Search Strategy of Intensification and Diversification in Tabu Search. *International Conference on Neural Networks and Signal Processing*, Nanjing, 14-17 December 2003, 428-431.