Application of Weighted Partial Max-SAT for Post-improvement of the Multiple Traveling Salesman Problem

João P. Lima[1], **Anne C. Rocha**[2], **Alexandre Arruda**[1], **Dmontier Jr.**[1]

[1]Universidade Federal do Ceará (UFC)
Russas, CE – Brazil

[2]Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP) – São Paulo, SP – Brazil

pedro34@alu.ufc.br, {alexandre.arruda,dmontier.aragao}@ufc.br

carolannie@ime.usp.br

Abstract. *Advances allow SAT solvers to be used to solve problems in the industrial sector. Therefore, in this paper, we reduce the multiple traveling salesman problem to a weighted partial max-SAT, with the aim of increasing the quality of the solution at a reduced computational cost. A version of Clarke and Wright's saving algorithm has been implemented to create the initial solution, while the 2-opt algorithm is applied to each route to improve the routes, the search space is extended by adding k nearest neighbors of each vertex so that post-improvement can be performed by the SAT solver. Benchmarks of four instances from the literature suggest a significant post-improvement in the quality of the solution up to* 43.51% *for a reasonable computational cost.*

1. Introduction

Boolean satisfiability (SAT) is a significant problem in both theoretical and practical terms [Sohanghpurwala et al. 2017]. The reduction of problems for SAT is characterized as a very successful approach to solving difficult combinatorial problems in artificial intelligence and computer science [Rintanen 2012]. According to [Rintanen 2012], reducing the problems for the SAT makes it possible to improve the problem-solving procedure.

The advances made in solvers with the insertion of new heuristics and increasingly refined implementations make it possible to obtain results for large and complex instances of combinatorial optimization problems; these advances allow the solvers to be used to solve problems in the industrial sector [Sohanghpurwala et al. 2017]. Motivating works such as the one presented in this paper and [Zha et al. 2020] to apply SAT solvers as a resource to solve the multiple traveling salesman problem (MTSP).

Considered one of the most interesting and popular problems in combinatorial optimization, MTSP is widely used to model and relax more complex problems, such as the vehicle routing problem [Cheikhrouhou and Khoufi 2021, Shokouhi rostami et al. 2015]. The complex nature of this problem makes it necessary to develop and apply optimization techniques to improve the quality of the final solution.

This paper modifies the approach proposed in [Rocha et al. 2023] to reduce the MTSP to a weighted partial max-SAT, with the aim of increasing the quality of the

solution generated at a reduced computational cost. In [Rocha et al. 2023] a post-improvement procedure was proposed for the traveling salesman problem that uses a reduction of the problem to a weighted partial Max-SAT. The authors used the solver to make additional improvements to the initial solution produced and improved by the 2-Opt algorithm. The main difference between the papers lies in the method used to create the initial solution and the logical formulation of the problem.

2. Literature Review

The Multiple Traveling Salesman Problem (MTSP) is a generalization of the well-known Traveling Salesman Problem (TSP), in which several salesmen are involved in visiting a given set of cities exactly once and returning to the starting position with the minimum travel cost [Cheikhrouhou and Khoufi 2021]. The travel cost metric can be defined in terms of distance, travel time, etc [Matai et al. 2010]. In this way, the MTSP is a relaxation of the VRP without considering vehicle capacity or customer demand [Cheikhrouhou and Khoufi 2021].

According to [Karabulut et al. 2021] MTSP can be formulated as follows: Let $G = (N, A)$ be a complete graph, where N is the set of n nodes and A is the set of arcs. Each arc $(i, j) \in A$ has a non-negative associated travel cost d_{ij}. Since there are m salesmen, the objective is to determine a tour for each salesperson in such a way that each tour starts and ends at the depot, at least one node is visited in each tour, and all nodes are visited only once by any salesperson. Then, the total cost of a tour v can be defined as C_v and the total cost of m tours can be defined as $TC = C_1 + C_2 + C_3 + ... + C_m$. Thus, the objective function of minsum MTSP is to minimize the total cost of m tours and can be described as $min(TC)$.

Being a combinatorial optimization problem, as the size of the graph increases, it becomes more challenging to discover optimal solutions due to the complex and non-deterministic nature of the problem. In addition, the computational cost of exploring all possible solutions in this search space can be extremely high and may even be unattainable based on the size of the problem in question, these characteristics of the problem allow for the implementation of approximate algorithms and heuristics [Tosoni et al. 2022]. A small improvement in the quality of the initially proposed solution or even a reduction in the execution time of the algorithm can result in significant savings in resources or in increased productivity of a company [Huerta et al. 2022].

2.1. Constructive methods

Being a subclass of the approximate algorithms, the tour construction algorithms build a solution by adding each city step by step, all tour construction algorithms stop when a solution is found and never try to improve it [Matai et al. 2010, Helsgaun 2000].

As a tour construction procedure, Clarke & Wright's Savings Algorithm was first proposed in [Clarke and Wright 1964] and is a widely known heuristic algorithm for applications in the traveling salesman problem (TSP) and the vehicle routing problem (VRP). The main objective of the algorithm is to assign routes to each salesman or vehicle based on the savings that are ordered in descending order.

A concise explanation of the algorithm as per [Segerstedt 2014]: Begin by computing a symmetrical distance matrix based on known distances. Then, determine the

distance savings for delivery from the depot for every possible pair of points on the same route. To initiate a new route, select the pair with the highest savings available (if the pair is still available for distribution). Once all distribution points have been assigned, the route is considered complete. Subsequently, include the distribution point that yields the greatest "total savings" for all distribution points already assigned to the route, considering the problem's constraints. If no additional distribution points can be accommodated, return to the initial step.

The nearest neighbor algorithm starts choosing any arbitrary node and adds the nearest node at each step until it has no more cities to include [Anbuudayasankar et al. 2014].

2.2. Improvement methods

After obtaining any solution generated by a walk construction algorithm, it is possible to add an improvement heuristic. Walk improvement algorithms aim to obtain an improvement in the cost of the initial solution by making several changes to the initial solution [Matai et al. 2010, Helsgaun 2000].

One of the main improvement methods applied to TSP is the 2-opt algorithm [Lin 1965]. The 2-opt algorithm randomly removes two non-adjacent edges from the tour and reconnects two new edges so that the tour remains a valid tour, the procedure is repeated until no improvement is possible, then the cost of the tour is reduced and the resulting tour is a valid tour. Furthermore, the quality of the improvement of the solution generated by the 2-opt algorithm is strongly dependent on the initial solution [Matai et al. 2010].

2.3. Composite methods

The tour construction algorithms gradually build up a tour by adding a new city at each stage. The improvement algorithms improve a tour by making several changes. Composite methods combine these two characteristics to generate an initial tour using a construction algorithm, and after obtaining the initial solution, one or more improvement procedures are applied in order to improve the initial solution given by the construction algorithm [Helsgaun 2000].

2.4. Logical approach

Let φ a propositional formula composed of a set of variables V, logical connectives $\wedge, \vee, \Rightarrow, \Leftrightarrow$ and $\neg$ (denoting conjunction, disjunction, implication, bi-implication, and negation, respectively) and parentheses. The formula φ is said to be satisfiable if and only if there is at least one valuation that satisfies it; otherwise, it is said to be unsatisfiable. In this way, SAT consists of finding a valuation that satisfies a given propositional formula [Vizel et al. 2015].

Each weighted clause in Max-SAT is a pair (c, W) where c is defined as a classical clause and w is a natural number that represents the cost of falsifying the associated classical clause. The clause is said to be hard if the cost associated with it is infinite, otherwise the clause is said to be soft. A weighted Max-SAT formula is a multiple set of weighted clauses $\phi = \{(C_1, w_1), ..., (C_m, w_m), (C_m + 1, \infty), ..., (C_{m+m'}, \infty)\}$, such that the first m clauses are soft and the last m' are hard [Ansótegui et al. 2010]

With ϕ being a multi-set of weighted clauses, the NP-hard problem of a weighted partial Max-SAT combinatorial nature consists of finding an optimal valuation for the clauses of ϕ such that it minimizes the cost associated with the valuation in ϕ. If the cost is infinite, it means that a clause called hard has been falsified, in which case we say that the multiset ϕ is unsatisfiable. Thus, the weighted Max-SAT problem is equivalent to the weighted partial Max-SAT when there are no hard clauses [Ansótegui et al. 2010].

Despite the combinatorial nature of the problem, with an exponential worst-case execution time. Modern SAT solvers use advanced branching heuristics, clause learning algorithms, and highly refined implementations. In this way, a large number of large and/or difficult instances of large problems (millions of variables and clauses) can be solved in execution times that are considered reasonable. These advances can be easily monitored by observing the results of the annual SAT Competitions. These advances in SAT solvers and their heuristics mean that many industrial problems can be solved quickly and efficiently [Sohanghpurwala et al. 2017].

3. Methodology

For the development of this work, comparative analyses were carried out with the sp11, uk12, eil51 and berlin52 instances. Instances sp11 and uk12 were obtained from `https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html`. eil51 was previously used by [Jiang et al. 2020] and berlin52 was ported from TSPLIB [Reinelt 1991]. For instances eil51 and berlin52 it was necessary to calculate the Euclidean distance between each city and build a distance matrix. Our experiments were run on AMD® Ryzen 7 6800H 4.7 GHz processor with 16 GB of 4800 MHz DDR5 RAM with Ubuntu 22.04.4 LTS operating system. The proposed approach can be consulted in the following repository with the possibility of replicating the experiments: `https://bitbucket.org/ltiufc/mtsp-modeling/src/main/`

3.1. Initial Solution

To build the initial solution, a version of Clarke and Wright's savings algorithm was implemented. Using as a restriction for joining routes the criterion that each salesman cannot visit a number of cities greater than the ratio of the number of cities to the number of salesman. In this way, the output of the algorithm is an initial solution of the routes for each salesman.

3.2. 2-opt Movement

Once the initial solution consisting of the route for each salesman has been obtained, the 2-opt improvement move is applied to each route individually by swapping adjacent edges until no further improvement is possible through these swaps.

3.3. Nearest neighbors

Next, the K nearest neighbors are added to each vertex of the path resulting from the improvement of the 2-opt algorithm, taking the proximity of the vertices as the main criterion of choice given the minimum spanning tree created from the distance matrix. In cases where the vertex does not have the K neighbors required in the configuration of the minimum spanning tree, the distance matrix is considered to choose the missing neighbors. Once the K neighbors have been added, a new distance matrix is generated which will be used as input in the logic modeling stage.

3.4. Logical Formulation

After adding K neighbors to each vertex, based on the dimensions of the distance matrix and a given number m of salesman, it is possible to define the clauses as follows. Each edge w that connects a city i to a city j in the format w_{ijv}, clauses that represent that it is true that the vehicle v passed through the edge (i, j) as c_{ijv}, and clauses t_{iv} representing the association of each vehicle v with a city i. Where $i, j \in \{0, ..., n\}$ given that $i \neq j$, and $v \in \{0, ..., m\}$, such that n represents the number of cities. So each clause can take on a true value (e.g.,$w_{ijv} = 1$) or a false value (e.g.,$w_{ijv} = 0$). In this way, each clause is mapped to an integer to build the model to be passed as input to the SAT solver.

The Nagoya Pseudo-Boolean Solver (NaPS) [Sakai and Nabeshima 2015] was chosen for this work due to its efficient and optimized implementation based on the underlying MiniSat, its exceptional results in SAT competitions, and its support for solving instances of the weighted partial Max-SAT with time limits.

After mapping the clauses, the model constraints and the minimization function 1 are specified for running the SAT solver. The constraints 2 and 3 ensure that only one edge exits the zero vertex for each salesperson, while 4 and 5 ensure that only one edge exits and enters each vertex.

$$Min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{v=0}^{m} w_{ijv} \times d_{ij} \tag{1}$$

$$\sum_{j=0}^{n} w_{ijv} = 1, \ \forall v \in \{0, ..., m\}, \ i = 0 \tag{2}$$

$$\sum_{i=0}^{n} w_{ijv} = 1, \ \forall v \in \{0, ..., m\}, \ j = 0 \tag{3}$$

$$\sum_{j=0}^{n} \sum_{v=0}^{m} w_{ijv} = 1, \ \forall i \in \{1, ..., n\}, \ i \neq j \tag{4}$$

$$\sum_{i=0}^{n} \sum_{v=0}^{m} w_{ijv} = 1, \ \forall j \in \{1, ..., n\}, \ i \neq j \tag{5}$$

$$-w_{ijv} - w_{jiv} \geq 1 \ \forall i \in \{0, ..., n\}, \ \forall j \in \{i+1, i+2, i+3, ..., n\}, \ \forall v \in \{0, ..., m\} \tag{6}$$

$$-w_{ijv} + c_{ijv} \geq 1, \ \forall i, j \in \{0, ..., n\}, \ \forall v \in \{0, ..., m\}, \ i \neq j \tag{7}$$

$$-w_{ijv} - c_{jkv} + c_{ikv} \geq 1, \ \forall i, j, k \in \{1, ..., n\}, \ \forall v \in \{0, ..., m\}, \ i \neq j \tag{8}$$

$$-w_{ijv} + t_{iv} \geq 1, \ \forall i, j \in \{1, ..., n\}, \ \forall v \in \{0, ..., m\}, \ i \neq j \tag{9}$$

$$-w_{ijv} + t_{jv} \geq 1, \ \forall i, j \in \{1, ..., n\}, \ \ \forall v \in \{0, ..., m\}, \ \ i \neq j \tag{10}$$

$$-t_{iv} - t_{il} \geq 1, \ \forall i \in \{1, ..., n\}, \ \ \forall v, l \in \{0, ..., m\}, \ \ v \neq l \tag{11}$$

$$-w_{ijv} + t_{iv} \geq 1, \ \forall i \in \{1, ..., n\}, \ \ \forall v \in \{0, ..., m\}, \ \ j = 0 \tag{12}$$

$$-w_{ijv} + t_{jv} \geq 1, \ \forall j \in \{1, ..., n\}, \ \ \forall v \in \{0, ..., m\}, \ \ i = 0 \tag{13}$$

The constraint 6 ensures that no vertex is visited more than once. Constraints 7 and 8 provide the path base and induction to avoid the formation of subtours in the solution. Constraints 9, 10, 12 and 13 link each salesperson to a city, and constraint 11 prevents a city from being visited by more than one salesman. After executing the SAT solver, the reverse mapping from integers to clauses takes place, in order to display the path obtained as a result, as well as the cost associated with the solution.

4. Results and discussions

To evaluate the impact of using the SAT solver on the final solution, the cost of the route was checked before and after post-improvement with the SAT solver, on four instances from the literature previously worked on in [Karabulut et al. 2021]. The quality criterion for the solution was the improvement made by using the solver compared to the initial solution, and the performance criterion was the average time of 10 executions of each instance due to random restarts of the solver.

The table 1 shows how close the cost found by the modeling is to the optimum cost for small instances; the costs differ from the optimum by 47.61% and 82.93% respectively for instances sp11 and uk12. Regarding the refinement of the modeling in relation to the initial solution, we can mention improvements of 43.51% for 11b with k = 5, 20.58% for 12b with k = 5 and 7.01% for berlin52 with five selesman and k = 5.

It can be seen that the use of the five nearest neighbors guarantees the greatest approximation to the optimum cost, as a result of the wider search space, allowing NaPS to reach solutions closer to the optimum in a reasonable execution time, as suggested by the results shown in the table. The berlin52 instance had the longest execution time for 3 salespeople and k = 5, although it was still less than 100 seconds, showing that the optimal cost was approached with minimal variation in the time required to apply the model.

The results suggest that the logical approach associated with constructive methods and improvement methods are efficient in obtaining near-optimal solutions with acceptable computational costs for small instances. It proves to be an appropriate approach for contexts in which approximations to the optimal value are accepted. The improvement achieved in the quality of the solution or in the computational cost required to obtain the route, however small, will still result in savings of millions of dollars for the company [Huerta et al. 2022].

In future work, we intend to investigate and analyze the impact of different strategies on the choice of K neighbors. Initial solutions construction strategies specific to the

MTSP will be explored, as a result of the possibility of increasing the quality of the solution space for solver exploration. In addition, to investigate anytime solvers with support for Weighted Partial Max-SAT and this work will be extended to the Vehicle Routing Problem.

Table 1. Benchmark results

Instances				Approach			
				C&W + 2-opt	Logical Approach		
Name	Size	Salesman	Optimal	Cost	Cost	K	Average Time (S)
					301	3	0.09
					215	4	0.12
11b	11	3	104	301	169	5	0.19
					3198	3	0.27
					3022	4	0.41
12b	12	3	1466	3543	2814	5	0.54
					721	3	3.51
					657	4	4.48
eil51	51	3	159.57	744	652	5	12.98
					13933	3	4.96
					12830	4	9.53
berlin52	52	3	7128	13933	12679	5	99.3
					977	3	5.48
					977	4	5.91
eil51	51	5	118.13	977	955	5	6.44
					19104	3	7.64
					18007	4	9.56
berlin52	52	5	7074	19104	17765	5	19.91

References

Anbuudayasankar, S. P., Ganesh, K., and Mohapatra, S. (2014). *Survey of Methodologies for TSP and VRP*, pages 11–42. Springer International Publishing, Cham.

Ansótegui, C., Bonet, M. L., and Levy, J. (2010). A new algorithm for weighted partial MaxSAT. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 3–8. AAAI Press.

Cheikhrouhou, O. and Khoufi, I. (2021). A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. *Computer Science Review*, 40:100369.

Clarke, G. and Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Huerta, I. I., Neira, D. A., Ortega, D. A., Varas, V., Godoy, J., and Asín-Achá, R. (2022). Improving the state-of-the-art in the Traveling Salesman Problem: An Anytime Automatic Algorithm Selection. *Expert Systems with Applications*, 187:115948.

Jiang, C., Wan, Z., and Peng, Z. (2020). A new efficient hybrid algorithm for large scale multiple traveling salesman problems. *Expert Systems with Applications*, 139:112867.

Karabulut, K., Öztop, H., Kandiller, L., and Tasgetiren, M. F. (2021). Modeling and optimization of multiple traveling salesmen problems: An evolution strategy approach. *Computers Operations Research*, 129:105192.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269.

Matai, R., Singh, S., and Mittal, M. L. (2010). Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches. In Davendra, D., editor, *Traveling Salesman Problem*, chapter 1. IntechOpen, Rijeka.

Reinelt, G. (1991). TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4):376–384.

Rintanen, J. (2012). Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193:45–86.

Rocha, A. C., Lima, J. P., Arruda, A., and Dmontier (2023). USING A MAX-SAT SOLVER AS A POST-IMPROVEMENT MOVE FOR TRAVELING SALESMAN PROBLEM. volume 2, page 908 – 916. Computers and Industrial Engineering.

Sakai, M. and Nabeshima, H. (2015). Construction of an ROBDD for a PB-Constraint in Band Form and Related Techniques for PB-Solvers. *IEICE Transactions on Information and Systems*, E98.D(6):1121–1127.

Segerstedt, A. (2014). A simple heuristic for vehicle routing – A variant of Clarke and Wright's saving method. *International Journal of Production Economics*, 157:74–79. The International Society for Inventory Research, 2012.

Shokouhi rostami, A., Mohanna, F., Keshavarz, H., and Rahmani Hosseinabadi, A. A. (2015). Solving Multiple Traveling Salesman Problem using the Gravitational Emulation Local Search Algorithm. *Applied Mathematics Information Sciences*, 9:1–11.

Sohanghpurwala, A. A., Hassan, M. W., and Athanas, P. (2017). Hardware accelerated SAT solvers—A survey. *Journal of Parallel and Distributed Computing*, 106:170–184.

Tosoni, D., Galli, C., Hanne, T., and Dornberger, R. (2022). Benchmarking Metaheuristic Optimization Algorithms on Travelling Salesman Problems. In *Proceedings of the 8th International Conference on E-Society, e-Learning and e-Technologies*, ICSLT '22, page 20–25, New York, NY, USA. Association for Computing Machinery.

Vizel, Y., Weissenbacher, G., and Malik, S. (2015). Boolean Satisfiability Solvers and Their Applications in Model Checking. *Proceedings of the IEEE*, 103(11):2021–2035.

Zha, A., Gao, R., Chang, Q., Koshimura, M., and Noda, I. (2020). CNF Encodings for the Min-Max Multiple Traveling Salesmen Problem. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 285–292.