Post-improving the Capacitated Vehicle Routing Problem Using a Max-SAT Solver

Pedro H. Ferreira[1], **Alexandre M. Arruda**[1]

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

pedrohferreira@alu.ufc.br, alexandre.arruda@ufc.br

Abstract. *SAT solvers are used to solve a wide variety of problems in artificial intelligence, especially the Weighted Max-SAT variation that contributes significantly to solving combinatorial optimization problems. In this paper, the capacitated vehicle routing problem is reduced to a weighted partial Max-SAT to improve solution quality with a reduced computational cost. An adaptation of Clarke and Wright's savings algorithm and the 2-opt algorithm have been implemented to construct the initial solution. The search space is expanded by adding the k-nearest neighbors to enable post-improvement by the SAT solver. Benchmarks of instances from the literature suggest a significant improvement in solution quality for a reasonable computational cost.*

1. Introduction

Boolean satisfiability (SAT) is the first problem that was proven to be NP-complete, and in addition it is also essential to artificial intelligence, algorithm and hardware design [Gong and Zhou 2017]. Reduction of problems to SAT is a very successful approach to solving hard combinatorial problems [Rintanen 2012]. Furthermore, due to annual SAT competitions, SAT solvers are constantly evolving to solve large instances of challenging problems in a reasonable time.

The weighted Max-SAT is the optimization version of the SAT problem and is also associated with problems in domains such as routing, bioinformatics, scheduling, probabilistic reasoning, and electronic markets [Larrosa et al. 2008]. In addition, the weighted Max-SAT problem has been extensively studied due to its applicability in diverse practical applications.

The vehicle routing problem (VRP) is one of the most important research topics in the field of logistics management and also belongs to a category of combinatorial optimization problems [Kao and Chen 2013]. This paper aims to improve the solving of the capacitated vehicle routing problem (CVRP), which is a variation of VRP that incorporates capacity constraints, making it more applicable to real-world scenarios. The nature of CVRP requires the development of optimization techniques to improve the quality of solutions.

In the literature, [Lima et al. 2024] proposed a post-improvement procedure for the multiple traveling salesman problem (MTSP), reducing the problem to a weighted partial Max-SAT. The authors created an initial solution for the problem using Clarke and Wright's savings algorithm and the 2-opt algorithm, then the solver was used to make additional improvements. The main differences between this paper and the work by

[Lima et al. 2024] lie in the problem addressed and the applied SAT solver, along with some variations in the Clarke and Wright's savings algorithm and in the logical modeling. The CVRP introduces vehicle capacity constraints, which increase the complexity of the problem by requiring solutions to simultaneously optimize route efficiency and satisfy load feasibility.

2. Literature Review

The vehicle routing problem (VRP) consists of planning routes for a fleet of vehicles in order to serve a given set of customer demands with the aim of minimizing the distance traveled [Khadilkar 2022]. Furthermore, it is very similar to the traveling salesman problem (TSP), the main difference is that TSP only produces a single route serving all customer demands using a single vehicle. In addition, CVRP is a more realistic version of VRP, which includes a capacity constraint on the total demand served by any given vehicle [Khadilkar 2022].

According to [Vidal 2022], the formal definition of CVRP is given as follows: Let $G = (V, E)$ be an undirected graph, where $V = \{1, \ldots, n\}$, with node 1 representing the depot from which a fleet of m vehicles operates, and the remaining nodes $\{2, \ldots, n\}$ representing customer locations. Each customer $i \in \{2, \ldots, n\}$ has an associated non-negative demand q_i, and each vehicle in the fleet has a fixed capacity Q. The set of edges is defined as $E \subseteq \{(i, j) \mid i, j \in V,\ i \neq j\}$, where each edge $(i, j) \in E$ is associated with a non-negative cost d_{ij} representing the distance between nodes i and j. The objective of CVRP is to create a partition of the set of customer nodes into m disjoint subsets, each representing a vehicle route that starts and ends at the depot and whose total customer demand does not exceed the capacity Q, while minimizing the total cost, measured as the sum of the distances traveled across all routes.

For instances with a large number of customers, finding an optimal solution becomes impractical due to the combinatorial nature of the problem. Exploring all possible solutions using exact methods is computationally expensive and, in some cases, infeasible [Tosoni et al. 2022]. In this way, the use of heuristics, metaheuristics, and approximation algorithms is an effective approach to address this challenge. Consequently, a small improvement in solution quality or at least a reduction in execution time can significantly increase the productivity of a company [Huerta et al. 2022].

2.1. Tour construction algorithms

Tour construction algorithms incrementally build a tour by adding one customer at a time, and the elaborated solutions offer a reduced execution time but with moderate quality [Helsgaun 2000, Rocha et al. 2023]. In this way, some of the most commonly used construction algorithms include the nearest neighbor algorithm and the Clarke and Wright's savings algorithm.

The nearest neighbor algorithm is easy to implement and works by starting any node as the beginning of a path, then adding the closest node to the most recently added node, continuing this process until all nodes are included in the path [Golden et al. 1980]. It does not guarantee an optimal solution, as its greedy approach may result in suboptimal paths.

The Clarke and Wright's savings algorithm, first proposed by [Clarke and Wright 1964], is a widely used heuristic to solve the vehicle routing problem (VRP). The goal is to minimize the total travel distance by iteratively merging routes based on the savings obtained by combining the customer pairs, leading to a more efficient solution. It starts by selecting any node as the central depot, which is commonly denoted as node 1. Subsequently, the savings $s_{ij} = d_{1i} + d_{1j} - d_{ij}$ are computed for $i, j = \{2, 3, \ldots, n\}$. The savings are then ordered in descending order. In the final step, starting at the top of the savings list and moving downward, larger subtours are formed by linking the appropriate nodes i and j. This process is repeated until no further improvements can be achieved by merging routes. A complete explanation of the algorithm was given by [Golden et al. 1980].

2.2. Tour improvement algorithms

Tour improvement algorithms begin with an arbitrary initial solution or one generated by a constructive algorithm, and iteratively perform various exchanges on the current solution to enhance the route, continuing until no further improvements can be made [Helsgaun 2000, Rocha et al. 2023]. The 2-opt algorithm is considered one of the most important improvement algorithms.

Initially introduced by [Croes 1958], the 2-opt algorithm is probably the best known heuristic for improving solution to the traveling salesman problem (TSP). It starts with a given tour, then replaces 2 links in the tour with 2 other links in such a way that the new tour length is shorter, continue in this way until no more improvements are possible. Furthermore, this process maintains the feasibility of the tour and involves reversing a subsequence of customers. A more detailed explanation is presented by [Helsgaun 2000].

2.3. Composite algorithms

The composite algorithms are the combination of the construction algorithms and the improvement algorithms [Helsgaun 2000]. Being relatively fast computationally and giving excellent results, the composite procedure aims to quickly obtain a good initial solution through construction algorithms, with the expectation that the improvement algorithms will subsequently refine it to an almost-optimal solution [Golden et al. 1980].

2.4. Logical approach

Boolean satisfiability (SAT) is a well-known combinatorial problem that determines whether a given circuit formula can be satisfied by a specific truth assignment, with a circuit instance represented in conjunctive normal form (CNF), which is a conjunction ($\wedge$ operator) of clauses, each being a disjunction ($\vee$ operator) of literals [Sohanghpurwala et al. 2017]. In addition, to satisfy the entire CNF instance, all clauses must be satisfied, which means that at least one literal in each clause should be evaluated as true.

Every weighted clause in Max-SAT is represented as a pair (C, w), where C is a classical clause and w is a natural number or infinity, indicating the penalty for falsifying C. A clause is considered soft unless its weight is infinity, in which case it is considered hard. A weighted partial Max-SAT formula consists of a multi-set of weighted clauses $\varphi = \{(C_1, w_1), \ldots, (C_m, w_m), (C_{m+1}, \infty), (C_{m+m'}, \infty)\}$, where the first m clauses are soft, while the last m' clauses are hard [Ansotegui et al. 2010].

Given a multi-set of weighted clauses φ the goal is to find an optimal assignment to the variables of φ that minimizes the cost of the assignment on it. If the cost is infinity, it means that we must falsify a hard clause and we say that φ is unsatisfiable. The weighted Max-SAT problem is a special case of the weighted partial Max-SAT problem when no hard clauses are present [Ansotegui et al. 2010].

This paper adopts the *clasp* solver, which is a conflict-driven answer set solver based on concepts from constraint processing (CSP) and satisfiability checking (SAT) [Gebser et al. 2007]. It was designed for optimal performance and built on solvers such as MiniSAT and is also capable of solving the weighted partial Max-SAT problem.

3. Methodology

In this work, comparative analyses were conducted with instances A-n33-k6, A-n37-k5, E-n31-k7, F-n45-k4, P-n101-k4 and P-n45-k5 from the CVRPLIB[1] library. In certain instances, the distance matrix was not provided, and as a result, the Euclidean distance between each pair of customers was computed to construct it. All experiments were executed on Ubuntu 22.04.5 LTS with an AMD Ryzen 7 5700X 3.40 GHz processor and 16 GB of 3200 MHz DDR4 RAM. The proposed approach, along with the replication instructions is available in a github repository[2].

3.1. Initial solution

To generate the initial solution, a modified version of Clarke and Wright's savings algorithm was implemented. Routes were merged based on saving values, ensuring that the total demand of each merged route did not exceed vehicle capacity constraints. This approach ensures that costs are consistently minimized while respecting capacity constraints. In this way, the output of the algorithm is an initial set of routes for each vehicle.

3.2. 2-opt movement

After the initial routes for each vehicle have been constructed, the 2-opt movement is applied to each route individually by iteratively swapping adjacent edges. This process continues until no further improvement can be achieved through these swaps. As a result, the algorithm effectively optimizes routes by eliminating crossings and minimizing the total travel distance.

3.3. Nearest neighbors

Subsequently, the improved routes, obtained through the 2-opt algorithm, are further enhanced by selecting the K nearest neighbors using the minimum spanning tree from the distance matrix as the primary criterion. In cases of missing neighbors, the distance matrix is used to identify the remaining neighbors. In addition, a new distance matrix is generated for each individual route, which is then used in the subsequent logic modeling stage.

[1] http://vrp.galgos.inf.puc-rio.br/index.php/en/

[2] https://github.com/PedroHenriqueFerreira/CVRP

3.4. Logical modeling

Once the K nearest neighbors have been added to each node, let d_{ij} be the Euclidean distance between customers i and j, q_i the demand of customer i, and Q the vehicle capacity. The boolean variables are then defined as follows. The variables w_{ijv} represent an edge connecting customer i to customer j using vehicle v, and t_{iv} indicates the assignment of vehicle v to customer i. Additionally, the variables u_{iv} denote the position of the customer i within the route assigned to vehicle v, and to encode u_{iv} in binary form, the variables u_{ibv} are introduced, following the equation:

$$u_{iv} = \sum_{b=0}^{\lceil \log_2(n-1) \rceil - 1} 2^b \, u_{ibv}$$

In this context, let $i, j \in \{1, \ldots, n\}$, with $i \neq j$, and $v \in \{1, \ldots, m\}$, such that n represents the number of customers and m represents the number of vehicles. Each variable can either be true (e.g.,$w_{ijv} = 1$) or false (e.g.,$w_{ijv} = 0$). Consequently, each variable is mapped to an integer, allowing the construction of the model that is subsequently passed as input to the SAT solver.

The *clasp* solver [Gebser et al. 2007] was selected for this work due to its highly efficient and optimized implementation, which ensures fast and accurate results even for complex problems. Additionally, *clasp* has demonstrated exceptional results in SAT competitions, which makes it particularly well suited for solving large-scale optimization problems.

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

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

$$\sum_{i=2}^{n} w_{ijv} = 1, \ \forall v \in \{1, \ldots, m\}, \ j = 1 \tag{3}$$

$$\sum_{j=1}^{n} \sum_{v=1}^{m} w_{ijv} = 1, \ \forall i \in \{2, \ldots, n\}, \ i \neq j \tag{4}$$

$$\sum_{i=1}^{n} \sum_{v=1}^{m} w_{ijv} = 1, \ \forall j \in \{2, \ldots, n\}, \ i \neq j \tag{5}$$

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

$$-w_{ijv} + t_{iv} \geq 1, \ \forall i, j \in \{2, \ldots, n\}, \ \forall v \in \{1, \ldots, m\}, \ i \neq j \tag{7}$$

$$-w_{ijv} + t_{jv} \geq 1, \ \forall i, j \in \{2, \ldots, n\}, \ \forall v \in \{1, \ldots, m\}, \ i \neq j \tag{8}$$

$$-t_{iv} - t_{il} \geq 1,\ \forall i \in \{2, \ldots, n\},\ \forall v, l \in \{1, \ldots, m\},\ v \neq l \tag{9}$$

$$-w_{ijv} + t_{jv} \geq 1,\ \forall j \in \{2, \ldots, n\},\ \forall v \in \{1, \ldots, m\},\ i = 1 \tag{10}$$

$$-w_{ijv} + t_{iv} \geq 1,\ \forall i \in \{2, \ldots, n\},\ \forall v \in \{1, \ldots, m\},\ j = 1 \tag{11}$$

$$u_{iv} - u_{jv} + (n-1)\, w_{ijv}\ \leq n-2,\ \forall i, j \in \{2, \ldots, n\},\ \forall v \in \{1, \ldots, m\},\ i \neq j \tag{12}$$

$$\sum_{i=1}^{n} q_i\, t_{iv}\ \leq Q,\ \forall v \in \{1, \ldots, m\} \tag{13}$$

Once the variables are mapped, the model constraints and the minimization function (1) are defined for the execution of the SAT solver. Constraints (2) and (3) enforce that each vehicle leaves and returns to the depot through a single customer, while (4) and (5) ensure that exactly one edge enters and exits each customer. Constraint (6) guarantees that no vehicle visits a customer more than once. Constraints (7), (8), (10) and (11) link vehicles to customers, while (9) establish that no customer is visited by more than one vehicle. Constraint (12) prevents subtour formation. The last restriction (13) enforces that each vehicle does not exceed its capacity. After running the SAT solver, integers are mapped back to variables to reveal the resulting path and its associated cost.

4. Results and Discussions

The impact of using the SAT solver on the final solution was evaluated by checking the route cost before and after post-improvement on six instances from the literature. The quality criterion was the improvement achieved by using the solver compared to the initial solution, while the performance criterion was the average execution time over ten runs of each instance due to the solver's random restarts. A time limit of 80 seconds was set for the solver to ensure consistent and fair comparisons between runs.

Table 1 presents the results of the proposed approach, which follows the methodology described in the previous section. Moreover, it incorporates an approach based on [Lima et al. 2024], employing the subtour elimination constraints specified by the authors. The proposed method outperforms the approach based on [Lima et al. 2024] in computational time in most instances, particularly for smaller ones, while maintaining equivalent solution quality.

The results demonstrate the proximity of the modeled costs to the optimal cost for smaller instances. Specifically, the model achieves refinements of 12% and 11% over the initial solution for instances A-n37-k5 and P-n101-k4, respectively. However, when the initial solution is already near optimal, the solver struggles to make further improvements.

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 *clasp*

to reach solutions closer to the optimum in a reasonable execution time, as suggested by the results shown in the table. The P-n101-k4 instance had the longest execution time using four vehicles and five neighbors, 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 indicate that the logical approach combined with constructive and improvement methods is effective in obtaining improvements with acceptable computational costs for small instances. It proves to be a suitable approach in contexts where approximations to the optimal value are acceptable. Even a small improvement in the quality of the solution or in the computational cost required to obtain the route can result in significant savings, potentially amounting to millions of dollars for the company [Huerta et al. 2022].

Future work will concentrate on examining a variant of the problem addressed, known as the Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW). Additionally, the research will be extended to investigate the impact of various strategies on the selection of K nearest neighbors, with the aim of improving the quality of the solution space to enable more effective solver exploration.

Table 1. Benchmark Results

Instances		Proposed Approach				Based on Lima et al.
		CW+2Opt	Logical Modeling			Logical Modeling
Name	Optimal	Cost	K	Cost	Time (s)	Time (s)
A-n33-k6	742	995	3	936	0.185	0.927
			4	916	0.258	0.982
			5	916	0.538	1.052
A-n37-k5	669	1160	3	1119	0.245	1.085
			4	1119	0.354	1.090
			5	1020	0.979	1.194
E-n31-k7	379	647	3	635	0.184	0.875
			4	635	0.203	0.895
			5	594	0.206	0.951
F-n45-k4	724	1292	3	1271	0.281	1.573
			4	1259	0.632	1.626
			5	1194	7.246	1.857
P-n45-k5	510	718	3	718	0.336	2.006
			4	690	0.692	2.030
			5	659	34.251	8.122
P-n101-k4	681	1162	3	1162	3.425	20.190
			4	1111	7.341	20.913
			5	1034	90.314	102.193

References

Ansotegui, C., Bonet, M. L., and Levy, J. (2010). A new algorithm for weighted partial maxsat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 3–8.

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.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Gebser, M., Kaufmann, B., Neumann, A., and Schaub, T. (2007). clasp: A conflict-driven answer set solver. In Baral, C., Brewka, G., and Schlipf, J., editors, *Logic Programming and Nonmonotonic Reasoning*, pages 260–265, Berlin, Heidelberg. Springer Berlin Heidelberg.

Golden, B., Bodin, L., Doyle, T., and Stewart, W. J. (1980). Approximate traveling salesman algorithms. *Operations Research*, 28(3-part-ii):694–711.

Gong, W. and Zhou, X. (2017). A survey of sat solver. *AIP Conference Proceedings*, 1836(1):020059.

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.

Kao, Y. and Chen, M. (2013). Solving the cvrp problem using a hybrid pso approach. In Madani, K., Dourado, A., Rosa, A., and Filipe, J., editors, *Computational Intelligence*, pages 59–67, Berlin, Heidelberg. Springer Berlin Heidelberg.

Khadilkar, H. (2022). Solving the capacitated vehicle routing problem with timing windows using rollouts and max-sat. In *Eighth Indian Control Conference*, pages 1–6.

Larrosa, J., Heras, F., and de Givry, S. (2008). A logical approach to efficient max-sat solving. *Artificial Intelligence*, 172(2):204–233.

Lima, J., Rocha, A., Arruda, A., and Jr., D. (2024). Application of weighted partial max-sat for post-improvement of the multiple traveling salesman problem. In *Proceedings of the 5th Brazilian Workshop of Logic*, pages 9–16, Porto Alegre, RS, Brasil. SBC.

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. *Computers and Industrial Engineering*, 2:908–916.

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.

Vidal, T. (2022). Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers Operations Research*, 140:105643.