A metaheuristic approach for an intermittent traveling salesperson problem extension

Pieter Leyman · Patrick De Causmaecker

Received: date / Accepted: date

Abstract We discuss the intermittent traveling salesperson problem (ITSP), an extension to the well-known traveling salesperson problem. Similar to the classical problem, the ITSP requires that a tour visits all nodes in the network, but each node now has a required processing time as well. Furthermore, the allowable consecutive processing time of a node is limited, which results in the introduction of waiting time and/or multiple visits. As a result, a valid tour for the ITSP may no longer constitute a Hamiltonian cycle, but could include loops as well as use edges multiple times.

We omit assumptions made in earlier work on the ITSP, to allow for a broader and more realistic discussion of the problem. Specifically, we generalize the underlying model that determines the maximum consecutive node processing time. The contribution of the research can be summarized as follows. First, we propose a metaheuristic algorithm for this extended ITSP. We focus on algorithm components and specifically propose several options for the decoding procedure from solution representation to an ITSP tour. Second, we perform computational experiments, to allow for meaningful insights into each algorithm component's performance. We generate sufficiently diverse test instances, to warrant generalizable conclusions.

Keywords Routing · Temperature functions · Metaheuristics · Solution representation

Pieter Leyman is a Postdoctoral Fellow of the Flemish Research Foundation with contract number 12P9419N.

P. Leyman
Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
E-mail: pieter.leyman@kuleuven.be

P. De Causmaecker
Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
E-mail: patrick.decausmaecker@kuleuven.be

1 Introduction

We consider the intermittent traveling salesperson problem (ITSP) as a variant of the traveling salesperson problem (TSP), in which each node may need to be visited more than once. Each node has a required processing time, and a maximum temperature is imposed above which overheating would occur (or, alternatively, a maximum node capacity would be exceeded). An important trade-off to be made is whether to wait for a node to cool down sufficiently to allow for further processing, or to move to another node, process (a part of) that one, and move back to the first node. Note that there is no restriction which states that one should go back to an earlier abandoned node as soon as possible. Depending on among others, the underlying graph structure, it may prove beneficial to only finish processing some nodes at a much later time.

A major assumption of earlier work by Pham et al. (2020) concerns the temperature increase and decrease functions. Whereas in earlier work, the node temperature is considered to increase and decrease in a linear fashion, we now wish to relax this assumption, and propose generalized temperature increase and decrease functions (Section 2).

Consider that, while we retain the original naming, specifically regarding temperature (functions), the ITSP can potentially be applied in many different fields. Examples are the processing of a metal surface with a laser, in which case the metal cannot be allowed to overheat, and the delivery of fuel to different locations, with both a fill and usage rate.

2 Problem definition

2.1 General

An undirected graph $G(V, E)$ can be used to model the ITSP, with V the set of vertices or nodes (including the depot), and E the edges between the nodes. Each node i has a required processing time p_i ($p_0 = 0$ for the depot) and a temperature τ_i which would be achieved if node i would be processed for p_i consecutive time units. The specific values for both p_i and τ_i are determined independently. The travel time between each pair of nodes (i, j) is d_{ij} ($d_{ij} \geq 0$). We assume that $d_{ij} = d_{ji}$ and that the triangle inequality holds, which means that the direct travel time between any pair of nodes is never larger than the travel time between both nodes via a detour to any other node. Furthermore, a maximum temperature of T_{max} is imposed. The goal of the ITSP is to minimize the time at which the "salesperson" returns to the depot and all nodes have been fully processed. As a result, the objective function value consists of the total processing time ($\sum_{i=1}^{|V|} p_i$), the times traveled between the nodes and any waiting time incurred. Consider that the latter two depend on the route selected through the network, whereas the former is independent of the route taken. Finally, we wish to point out the use of some terminology. For the ITSP, each *node* i may be *visited* multiple times, with a different visit implying

that at least one other node has been (partially) processed and traveled to in between. Additionally, each *visit* may consist of multiple *processing steps*, which are separated by waiting times. In this case, however, we do not leave node i, but wait for it to cool down, before we continue with processing.

2.2 Temperature functions

To model the temperature $T_i(t)$ of node i after t_i consecutive time units of processing we use the following general polynomial function:

$$T_i(t_i) = \tau_i \cdot (t_i/p_i)^a,\ t_i \leq p_i \tag{1}$$

with a a constant which determines the increase rate ($a \in]0; +\infty[$). A value of a between 0 and 1 (neither included) leads to a concave temperature increase function (fast initial increase), whereas a value larger than 1 leads to a convex function (slow initial increase). A value of 1 corresponds with a linear increase. Figure 1 shows an example of how the node temperature increase function can differ based on the selected value for a. Furthermore, a can be seen as a feature of the problem studied and the nature of processing, which determines how the temperature of the material increases.

Allow us to illustrate our reasoning with a simple example. Assume that for a given node i ($p_i = 10, \tau_i = 10$), we have a value for a of 0.5 (which corresponds with the $a = 0.5$ curve in Figure 1). If we furthermore assume a T_{max} value of 7, this leads to a maximum consecutive processing time c_i for node i of 4 time units ($= \lfloor p_i \cdot \exp(\ln(T_{max}/\tau_i)/a) \rfloor$), by rounding down the inverse of Function (1) given a temperature $T_i(t)$ of 7. Note that we round the result down to the nearest integer, since we assume integer processing times.

So far, however, we have only explained how the temperature increase is modeled, but not the temperature decrease. Provided that the increase function, given a node i, only depends on the heating parameter a, we choose to employ the same function but with a different cooldown parameter b:

$$T_i(t_i) = \tau_i \cdot (t_i/p_i)^b,\ t_i \leq p_i \tag{2}$$

The way we want to use this decrease function is, however, different from the manner in which we use the increase function. Recall from the earlier example, that for node i we had determined that the maximum number of consecutive time units of processing c_i was 4. The cooldown function now allows us to determine how long it takes to once again reach the node start temperature of 0, assuming no further processing occurs in the meantime. From Function (1) we calculate the actual temperature after processing 4 time units, which yields 6.32. This actual temperature will never be higher than T_{max}, since c_i is determined by rounding down the inverse of Function (1), given T_{max}. We now calculate the inverse of Function (2), assume $b = 2$, with a temperature of 6.32. We round up the resulting number of time units of 8.37 to 9, to ensure that node i fully cools down.

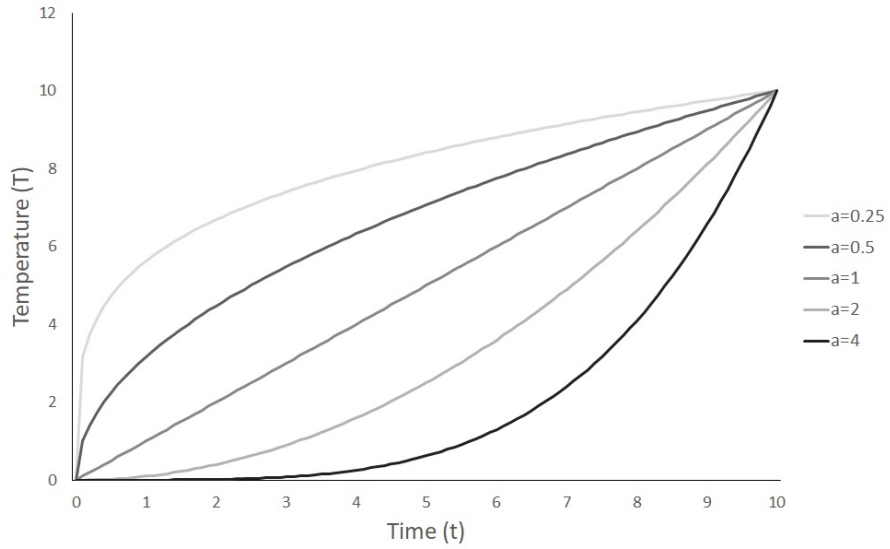

Fig. 1 Example temperature functions: the higher (lower) the value for a, the slower (faster) the initial T increase.

Table 1 Example of how a visit to node i can be split into multiple processing steps (PS), with waiting time in between the different steps to ensure node i cools down sufficiently.

PS	1	2	3
t_i	4	4	2
w_i	9	3	0

Recall, however, that the total p_i value of node i in the example was 10. One way to finish work on node i entirely is to process another 4 time units, cool down such that the final 2 time units can be processed, and then finish processing. For the example in Table 1, this leads to another waiting time of 3 time units, based on functions (1) and (2) (details are omitted for the sake of conciseness).

A summary of the way in which node i is processed in the example is given in Table 1, with t_i the selected processing time for the current processing step PS and w_i the waiting time after processing. Terminology wise, it is worth mentioning that in this example we use three *processing steps* for a single *visit* to *node* i.

The manner in which Functions (1) and (2), based on given a and b values, are employed, can then be summarized as follows:

1. Function (1) is used to model the temperature increase of a node i.
2. The maximum number of consecutive time units being processed (c_i) is calculated, based on the inverse of Function (1) and a given value for T_{max}.

3. Based on a processing time t_i, which is never larger than c_i, the resulting temperature is determined by Function (1).
4. This temperature value is then employed to calculate the required cooldown time to reach a node temperature of 0, based on the inverse of Function (2).
5. The interplay between Functions (1)-(2), and hence between a and b, determines what the best way is to minimize waiting time, for a given visit of a node i and a required processing time for that visit.
6. Different values for a and b allow for vastly different types of increase and decrease functions, and hence different types of applications.

3 Methodology

3.1 Metaheuristic framework

We employ the metaheuristic framework of Vidal et al. (2012, 2013) as starting point for our own algorithm. The hybrid genetic search with adaptive diversity control (HGSADC) metaheuristic is a hybrid metaheuristic framework, which combines the exploration elements of a classical genetic algorithm with both population diversity management and with efficient local search procedures. Given the intricate nature of the ITSP as discussed in Section 2, we have selected the HGSADC as metaheuristic framework, since it allows for a focus on local search techniques and deals with infeasible solutions, both of which we feel are crucial for the ITSP. Furthermore, the HGSADC achieved excellent results for the vehicle routing problems discussed in Vidal et al. (2012, 2013). Finally, in this abstract we focus, albeit briefly, on the components that we designed specifically for the ITSP, namely the solution representations and decoding procedure, since we have not deviated from the general structure of the HGSADC. An overview of the decoding procedure is provided in Figure 2.

3.2 Decoding procedure

Since a metaheuristic algorithm operates on a solution representation and not on a solution itself, this is a crucial algorithm component. Leyman et al. (2019) unambiguously showed that, albeit for a specific project scheduling problem, the choice of a solution representation deserves more attention than it in general receives in literature. The following two types of solution representations are used:

- Node list + processing time list (NL+PTL): A NL holds the order in which the nodes are to be visited, similar to a representation for the TSP, with the major difference that nodes may occur more than once, signifying multiple visits. The PTL holds the time processed during the current visit to the corresponding node in the NL (both the NL and PTL have the same length). A major downside of this PTL is that the sum of all values for a

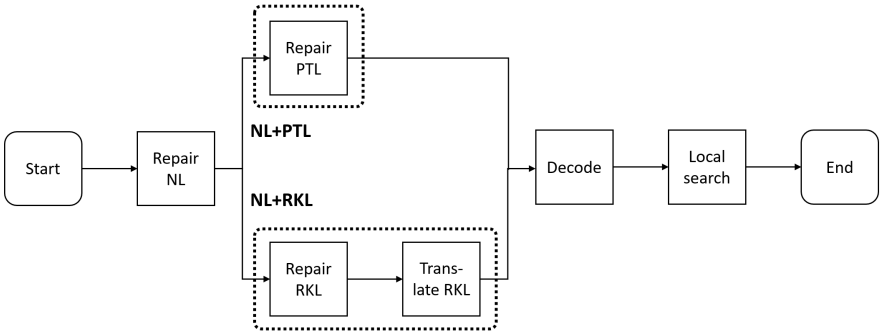

Fig. 2 Overview different steps decoding procedure.

given node i not only has to equal p_i, but also that each PTL value has to lie in the interval $[1; p_i]$. As a result, a repair operator is required as well.
- Node list + random key list (NL+RKL): The NL functions in the same manner as discussed earlier, but the general logic behind the RKL is that it contains only a floating point number between zero and one for each visit. Specifically, each value in the RKL corresponds with the fraction of the node's processing time to be completed during the current visit. This way, we only need to make sure that the sum of the RKL values for the same node equals one. We do, however, need a translation procedure when constructing a tour, to convert the RKL values to actual processing times.

4 Computational study

4.1 Test data

We generate test data based on the features in Table 2. Aside from the temperature constrainedness (TC), all features have been discussed in Section 2. The value for TC determines how tight the maximum temperature restriction is, given τ_i values for individual nodes. The value for T_{max} is calculated as follows: $T_{max} = \sum_{i=1}^{|V|} \tau_i \cdot$ TC $/|V|$. As a result, a low (high) value for TC leads to a low (high) T_{max} value as well. For each combination of instance features, 10 instances are generated, which results in a total dataset of 4320 instances.

4.2 Computational results

Table 3 shows a summary of the results for both the NL+PTL and NL+RKL combinations in terms of total time, total travel time and total waiting time. Each time, a stopping criterion of 5000 tours was used, to allow for a computer and code independent comparison (Leyman and De Causmaecker, 2017). Each cell in the table contains the average value over all 4320 test instances used.

Table 2 Instance features and feature values of test data.

Feature	Values
$\|V\|$	50, 100
p_i	$[1;20]$, $[1;100]$
τ_i	$[1;20]$, $[1;100]$
d_{ij}	$[1;20]$, $[1;100]$
TC	0.25, 0.5, 0.75
a	0.5, 1.0, 2.0
b	0.5, 1.0, 2.0

Table 3 Overview results for different solution representations (average).

	Total time	Total travel time	Total waiting time
NL+PTL	5030.93	1155.27	1567.95
NL+RKL	5036.09	1160.86	1567.51
Pham et al. (2020)	5647.83	1227.98	2112.13

Table 4 Temperature function results for different solution representations (average total time).

		NL+PTL	NL+RKL	Pham et al. (2020)
a	0.5	**7234.18**	7245.81	8587.31
	1	**4147.95**	4158.06	4467.74
	2	3710.66	**3704.40**	3888.44
b	0.5	3976.68	**3967.76**	4123.68
	1	**5050.12**	5057.89	5484.14
	2	**6066.00**	6082.62	7335.66

We also compare with the results of Pham et al. (2020), by including their decoding procedure in the HGSADC. This way, we can show that there is a clear contribution of taking the temperature functions into account in the proposed decoding procedures, since Pham et al. (2020) assume linear functions and do not include any mechanism to account for different types of functions.

Based on the results in Table 3, we can conclude that both NL+PTL and NL+RKL outperform the other case, but that the difference between themselves is small. The large improvement in average waiting time compared with Pham et al. (2020) shows that explicitly considering the temperature functions is worth its salt.

Table 4 provides more detailed results of the three alternatives, by splitting results based on the different a and b values used in the test design (Table 2). In general, the results are in line with those of Table 3, but we can see that regardless of a and b values there is an improvement compared to the decoding procedure of Pham et al. (2020). Hence, we can argue that our decoding procedure performs better in general, since linear functions are included in the results of Table 4 as well ($a = 1$, $b = 1$).

4.3 Critical remarks

Due to the results discussed above, we should more closely study the impact of travel versus processing times of a visit, since currently it seems that the algorithm prefers waiting time over travel time. We believe this is due to the travel times in the test data on average being of similar size as the processing times. This implies that if the average travel times are smaller than or similar to the average processing times, it is preferential to wait instead of moving to another node. We assume that as the travel times grow comparatively smaller, moving to another node instead will become the better choice, on average. But in order to be able to verify (or reject) this assumption, further testing with a more diverse test set is required.

5 Conclusions & future work

In this paper, we have discussed the intermittent traveling salesperson problem (ITSP) with generalized temperature functions. The ITSP is an extension of the well-known traveling salesperson problem (TSP), but in which each node has to be processed for a given duration and not just visited. Processing a node increases the node temperature, which should not exceed a maximum value. As a result, the ITSP may require multiple visits to nodes, unlike the TSP, where a single visit is imposed. Additionally, we have proposed general polynomial functions, which determine the manner in which node temperature increases and decreases.

To solve the ITSP with generalized temperature functions, we have employed the hybrid genetic search with adaptive diversity control of Vidal et al. (2012, 2013) as metaheuristic framework, and focused our contribution on solution representation alternatives and a decoding procedure.

In the future, we will study specific applications of the ITSP, closely linked to real-world problems to demonstrate the ITSP's practical relevance. Alternatively, we aim to consider heat transfer functions from physics and thermodynamics, which differ from the presently modeled cooling schemes for e.g. the metal surface processing application discussed earlier.

From a methodology and testing point of view, we will further investigate the decoding procedures, to better grasp the ITSP's intricacy, and to include additional solution representations. The test design should also be expanded to allow for networks of different (and larger) sizes, alongside a greater degree of variation in the heating and cooldown feature values. The algorithm's parameters should be tuned using an automatic algorithm configurator (e.g., SMAC, ParamILS, irace) rather than using the default values of Vidal et al. (2012, 2013) and ad-hoc values for additional parameters. Finally, the results should be analyzed in more detail, to allow for a clear evaluation of the contribution of each algorithm component, and to determine for which (type of) instances which solution representation leads to the best results.

References

Leyman P, De Causmaecker P (2017) Evaluation of metaheuristics: Is computation time worth the time? In: Proceedings of the 31st annual meeting of the Belgian Operational Research Society (ORBEL), pp 89–90

Leyman P, Van Driessche N, Vanhoucke M, De Causmaecker P (2019) The impact of solution representations on heuristic net present value optimization in discrete time/cost trade-off project scheduling with multiple cash flow and payment models. Computers and Operations Research 103:184–197

Pham TS, Leyman P, De Causmaecker P (2020) The intermittent travelling salesman problem. International Transactions in Operational Research 27:525–548

Vidal T, Crainic T, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60(3):611–624

Vidal T, Crainic T, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers and Operations Research 40:475–489