# Online Heuristic for the Multi-objective Generalized Traveling Salesman Problem

Joost van Pinxten*, Marc Geilen*, Twan Basten*‡, Umar Waqas*, Lou Somers†
*Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
†Research and Development, Océ Technologies, Venlo, The Netherlands
‡TNO-ESI, Eindhoven, The Netherlands
Corresponding author email: j.h.h.v.pinxten@tue.nl

***Abstract*—Today's manufacturing systems are typically complex cyber-physical systems where the physical and control aspects interact with the scheduling decisions. Optimizing such facilities requires ordering jobs and configuring the manufacturing system for each job. This optimization problem can be described as a Multi-Objective Generalized TSP where conflicting objectives lead to a trade-off space. This is the first work to address this TSP variant, introducing a compositional heuristic suitable to online application.**

## I. Introduction

The Traveling Salesman Problem (TSP) is a combinatorial optimization problem that aims to optimize a tour, visiting each of a set of cities exactly once. Many scheduling problems requiring sequence optimization can be defined as a variant of the TSP [1]. We explore online optimization techniques for scheduling Flexible Manufacturing Systems (FMS) that besides sequencing, involve selection optimization for multiple objectives. These can be expressed as a Multi-Objective Generalized TSP (MO-GTSP). An example FMS is a high performance production printer. Jobs from a print pool are executed in one of several modes, resulting in different processing values (time, quality, etc.). Changing modes occurs only between jobs. The goal is to optimize over the job sequence as a whole. Mode switches typically have a processing penalty, for instance time and/or cost. A scheduler needs to determine in what order to execute the jobs, and in which modes. Fig. 1 gives an example encoded in a MO-GTSP.

Optimization problems are often multi-objective in nature. The goal of (meta-)heuristics for multi-objective optimization problems is to find the efficient set, a set of solutions with Pareto optimal values. The efficient set consists of all solutions that have values that are not dominated by any other solution. The Multi-Objective Traveling Salesman Problem (MO-TSP) is the variant of the TSP in which multiple objectives, i.e. tour-metrics, need to be optimized simultaneously.

Many problems are also multiple-choice in nature; we have to select, for example, exactly one printer mode per print job. Choices occur in reconfigurable FMS that can be optimized for input sequences, or where the input sequences can be adapted to increase performance. This freedom of choice is captured by the generalized aspect of the Generalized TSP (GTSP), where an optimal tour through clusters of cities needs to be found, visiting exactly one city from each cluster.

Not all information may be available before any decisions need to be taken. For a printer, jobs can arrive at any point in time. An algorithm needs to be fast enough to recompute all optimization decisions, or to have an online component, where information can be incorporated on-the-fly.

The state-of-the-art algorithms for solving MO-TSP are non-deterministic meta-heuristics with powerful local search. For GTSP, the state-of-the-art consists of memetic algorithms (also known as genetic local search algorithms). Neither of these heuristics are fast enough for online optimization. MO-TSP solutions cannot be trivially extended to efficiently incorporate the multiple-choice aspect of GTSP. Similarly, single-objective optimization heuristics cannot be trivially extended to efficiently deal with multiple objectives. We are the first to develop a dedicated heuristic for the MO-GTSP. Our heuristic is compositional in nature, enabling online application.

The Compositional Pareto-algebraic Heuristic (CPH) [2] [3] is a heuristic approach that has been specifically designed for multi-objective, multiple-choice optimization problems. CPH has been applied successfully to the Multi-dimensional Multiple-choice Knapsack Problem (MMKP) [3]. Such problems are found in e.g. online resource management and in routing of wires on chips. The compositional and parametrized nature of CPH has shown particularly flexible and powerful for use as a tunable, online algorithm. The main difference between MMKP and MO-GTSP is that MMKP is essentially a selection problem whereas MO-GTSP is a combined sequencing and selection problem, and that TSP does not consider resource bounds. We use CPH for MMKP as a basis for our MO-GTSP heuristic.

Our target application domain is FMS. These systems are complex Cyber-Physical Systems that convert print jobs into printed documents such as books and high volume personalized transaction printing, at rates of hundreds of A4s per minute. As such, we are concerned with online computation of results.

## II. Related work

Although many algorithms have been developed for many TSP variants, like the GTSP and MO-TSP, no algorithms dedicated to solving the combined challenge of GTSP and MO-TSP have been published yet. We discuss work related to MO-TSP, GTSP, and online algorithms for TSP.

Many heuristics have been defined for MO-TSP. Stochastic Local Search (SLS) [4] is among the state-of-the-art MO-TSP meta-heuristics. Heuristics such as SLS combine non-deterministic search with powerful local search heuristics. Iterated Local Search (ILS) heuristics [5] search the neighbourhood to find a local minimum, and escape local minima by applying perturbations. SLS uses ILS to quickly search through the neighbourhood of the previously generated solution to find another solution that is likely in the efficient set. The heuristic sweeps over the objective space by gradually modifying the local search direction.

Memetic algorithms are regarded as the state-of-the-art heuristics for GTSP [8]. Memetic algorithms combine the ideas of crossover and mutation from genetic algorithms with a strong local improvement element. Memetic algorithms differ from SLS as they keep more than one solution, and recombine the good parts of different tours into a new tour. Memetic algorithms are typically slower than their deterministic counterparts, but have more opportunities to escape local minima.

The online TSP is typically generalized to a Dynamic Vehicle Routing Problem (DVRP) [6]: cities become available

This research is supported by the Dutch Technology Foundation STW under the Robust CPS program (project 12693)

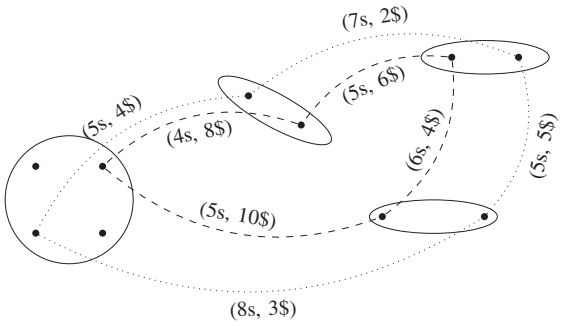



Fig. 1: Example MO-GTSP, with time and cost objectives. The ellipses with dots are clusters of cities in the MO-GTSP, representing print jobs and printer modes, resp. The two tours, print sequences and mode selections, have objective values (25s, 14$) and (20s, 28$) resp., showing a time, cost trade-off.

over time, and all the cities need to be covered with a tour by a limited number of vehicles. An extensive survey [7] of the multi-objective variant of DVRP shows that many variants exist with different objectives, on which many different techniques have been applied. The GTSP is for our purposes more relevant than the DVRP; the multiple-choice aspect of GTSP has not received much attention in the DVRP.

None of the above approaches can be easily combined or extended to tackle the online MO-GTSP; we extend the state-of-the-art memetic algorithm [8] to multiple objectives in Section V-B. We combine the compositional approach of CPH with powerful local search heuristics for TSP and GTSP such as used in ILS and the memetic algorithms into a new online MO-GTSP heuristic.

## III. The Multi-Objective Generalized TSP

We extend the single-objective GTSP model presented in. [9] to a MO-GTSP: We are given a complete directed graph $G(V, E)$ with vertices $V$ and edges $E = V \times V$. In addition, a proper partitioning of $V = C_1 \cup C_2 \cup ... \cup C_m$ defines the clusters to be visited. A cycle is *feasible* if it visits each cluster *exactly* once. A cost function $c : E \to \mathbb{R}^n_+$ maps each edge to its corresponding $n$-dimensional cost. Solutions to the problem are tours $T \subseteq E$, which are feasible cycles with Pareto-optimal multi-objective tour cost $\sum_{e \in T} c(e)$. The sum of value-tuples is obtained through element-wise addition.

A tour $T$ is *dominated* by some other tour $T'$ iff $T_i \geq T'_i$ for all dimensions $i$, i.e. iff each objective is worse or equal to that of that other tour. The Pareto-optimal tours are those that are not dominated by any other tour. The two tours in Fig. 1 are not dominated by each other.

## IV. A Compositional Pareto-algebraic Heuristic for Multi-Objective Generalized TSP

CPH has been successfully applied to MMKP [3], a multi-objective, multiple-choice problem like MO-GTSP. CPH is based on Pareto Algebra [10] and the idea that partial solutions can be computed incrementally; i.e. composing the final solutions by iteratively considering the combination of partial solutions. In this paper we refer to CPH for MMKP as CPH-MMKP, and CPH refers to the framework defined in this section. Our application of CPH to MO-GTSP is referred to as CPH-GTSP.

### A. Compositional Pareto-algebraic Heuristic

The template for a CPH-based algorithm is shown in Algorithm 1. It abstracts from the particular problem definition, and shows how multi-objective optimization problems can be tackled. The basic idea is that Pareto-optimal solutions are built by compositionally combining partial solutions. During the composition steps, at most $k$ solutions are maintained to restrict growth of the search space.

**Algorithm 1** Generic CPH framework

**Require:** problem instance, max. number $k$ of intermediate results
  **Step 1**: Initialize set of partial solutions $S_p$ and set of partial solution sets $S$
  **Step 2**: Perform compositional computations
  **for all** $S_i \in S$ **do**
    // reduce the set of solutions to $k$ elements
    **a**: $S_p$ = Reduce($S_p$, $k$)
    // combine two sets of partial solutions
    **b**: $S_p$ = Combine($S_p$, $S_i$)
    // Pareto-filter the new set of partial solutions
    **c**: $S_p$ = Pareto-minimize($S_p$)
  **end for**
  **Step 3**: **return** solutions $S_p$

*Step 1* initializes the partial solutions $S_p$ that will develop into the final solutions, i.e. the Pareto-optimal solutions to the problem instance. It also initializes the set of partial solutions to be considered in the compositional computations. Then we start combining sets of partial solutions in *Step 2*. This step is executed once for each partial solution set, incrementally composing the final solutions. *Step 2a* ensures that no more than $k$ elements are kept in $S_p$. *Step 2b* combines two sets of partial solutions to create a new set of partial solutions. From the resulting solutions, we only keep those that are non-dominated by applying Pareto-minimization at the end of each iteration (*Step 2c*). Once all sets are combined, the results are returned in *Step 3*.

*Steps 2b and 2c* can be combined in an implementation to increase the computational efficiency of the heuristic. Pre- and post-processing steps can be introduced to tune the search direction of the algorithm. In addition, during the compositional phase, a quick, typically greedy, improvement heuristic can be applied after *step 2c* to improve partial solutions.

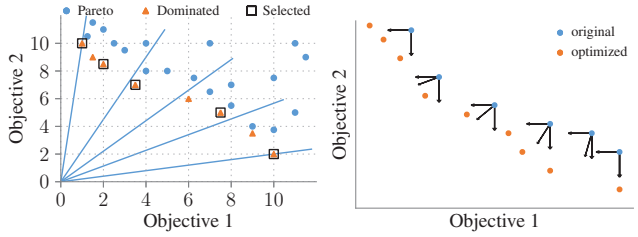
### B. CPH *applied to* MO-GTSP

This subsection details the operators that we use to allow MO-GTSP instances to be solved by CPH-GTSP.

*1) Initialization of partial solutions:* We initialize the set $S$ by creating sets of single-vertex tours. Each cluster is translated to a set of such partial tours in $S$. The set of partial solutions $S_p$ initially contains no elements.

*2) Reduction operator:* In *Step 2a* the set of partial solutions is reduced to size $k \geq 2$. The reduction mechanism in CPH is used to control the size of the search space, and therefore the execution time. As described in [3], this can be done in several ways. For two-objective problems, we use a selection mechanism that slices the 2-D space into equal parts and selects one solution per slice. Fig. 2a shows how the space between the two extreme Pareto values is subdivided by $k - 2$ lines with equal angles between subsequent lines. The operator returns the extreme points and one of the solutions per subdivision.

*3) Combination operator:* CPH computes the final results in a compositional manner; the algorithm builds up sets of useful partial solutions gradually in $S_p$ and combines these sets with other sets of partial solutions $S_i$ to end up with the final set of solutions. The combination operator that we employ exhaustively explores all combinations of the two partial solutions. In the simplified case, it means that per sub-tour $T$, $|T|$ new partial solutions are created in $S_p$ by inserting the single vertex solutions from $S_i$ into any of the positions of sub-tour $T$. Combining two optimal sub-tours in this way does not necessarily generate the optimal sub-tour, as is shown in




(a) Reduction mechanism. (b) Improvement vectors.

Fig. 2: Examples of operators.

Fig. 3 where the goal is to minimize the tour length. It shows an example where the best possible combination is not an optimal tour.

*4) Pareto minimization:* The solutions generated by the combination operator may not all be Pareto-optimal. We can remove all dominated solutions from the solution set by applying Pareto minimization. This minimization process removes all dominated solutions and is typically implemented by the *simple cull* algorithm as described extensively in [3] and [11].

*5) Improvement operators:* The basic idea of improvement operators is that they can reconsider previous choices which may have led to sub-optimal results. Such sub-optimality may be introduced by a heuristic combination operator, by the reduction operator, or by previous optimizations. We use the weighted sum schema [12] (i.e. a linear combination of all objectives) so that single-objective local optimization techniques can be used to push the value of a solution into a certain direction as illustrated in Fig. 2b.

As indicated by the current state-of-the-art [4], [9], [8], 2-opt exchange and 3-opt exchange neighbourhoods can be applied efficiently and reorder the solutions effectively to improve the results. These exchange operators cut the tours into two or three parts respectively and join them together again in a different arrangement. We use the 2-opt and reduced 3-opt exchange neighbourhood as described in ILS [5]. We do not use the full 3-opt, as any reversed part of a tour is unlikely to yield improvements after it has been optimized with a 2-opt.

In addition to the 2-opt exchange and 3-opt exchange neighbourhoods, we have adopted the *Cluster-Optimization* heuristic introduced in the branch-and-cut approach [9]. Given a fixed cluster order, it optimizes the selection of cities per cluster through a modified shortest-path algorithm. This heuristic has been shown to explore a very large neighbourhood [13] in polynomial time and is typically very fast. We always apply cluster optimization as the last improvement operator. The 2-opt and 3-opt exchange operators improve the ordering of the clusters, while the cluster optimization improves the selection of the nodes inside the cluster. Together, they effectively solve the simultaneous sequencing and selection problem of GTSP.

## V. Benchmarks and Experimental Evaluation

We assess how CPH performs as an offline and online MO-GTSP heuristic by comparing the runtimes and solution quality of CPH with the state-of-the-art. As no algorithm has been published for MO-GTSP we extend the current state-of-the-art GTSP to multiple objectives, to create a reference.

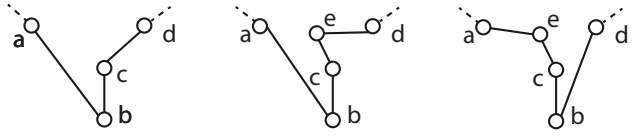



Fig. 3: (left) Optimal partial tour, (middle) best-insertion of new vertex $e$, (right) optimal tour including vertex $e$.

We set up several experiments to assess the performance of CPH-GTSP relative to the state-of-the-art Memetic Algorithm for GTSP (MA-GTSP) in both runtime and solution quality. To do so, we first present the benchmark and the details of the reference algorithm. We then present our method to compare the quality of the calculated Pareto fronts and finally perform the experiments that show the offline and online performance of CPH-GTSP compared to the state-of-the-art.

### A. MO-GTSP *benchmark*

We consider two benchmarks; an extension of the de-facto standard GTSPLIB benchmark and one based on FMS.

*1) Krolak instances from GTSPLIB:* The GTSP Instances Library[1] (GTSPLIB) provides a subset of instances from the TSPLIB with cluster information [8]. The Krolak instances in this set represent distance problems with geographical clustering. A single-objective GTSP definition in the GTSPLIB consists of two parts; the distances between nodes, and the clustering data. Multi-objective instances are obtained by using the distance data of two or more instances as the objectives, and the clustering data of one of the instances.

We combine five instances (20KroA100 to 20KroE100), each containing 20 clusters, and two instances with 40 clusters (40KroA200 and 40KroB200) to form 22 bi-objective MO-GTSP instances. We adopt the notation used for both MO-TSP and GTSP as follows: 20KroAB100 denotes the instance with 20 clusters, where the vertices and edge values are defined by the 20KroA100 and 20KroB100 instances, and the clusters are defined by the 20KroA100 instance.

*2) Flexible Manufacturing Systems benchmarks:* Subsequent print jobs interact with each other, as they can be (partly) interleaved when doing duplex printing in a printer with a large duplex loop [14]. Especially when many small jobs are processede, interleaving them benefits the total productivity. The interleaving options can be limited by the particular sequence of print media, and the printer and job configurations.

The flow-shop instances and heuristics of [14] have been used to determine the makespans of jobs and setup times between jobs of a printer. By changing some of the input parameters for these jobs (e.g. paper velocity, deviation in thicknesses), we have generated alternative job configurations. With these results, we have synthesized a new set of GTSP benchmark instances that show structures we expect to see when modelling the productivity and quality aspects of a print production system.

The setup times of the printer depend on the current mode, and impact the overall productivity. This generates inherent trade-offs between optimized jobs and optimized transitions. Additionally, deviations in the jobs allow subsequent jobs to be executed with reduced setup times. As such deviations may be less desirable, a second objective is created from the sum of the squares of the normalized deviation values per job as the quality penalty. A deviation depends only on a single job, and is accounted for in the outgoing edges from the job. We aim to minimize quality penalty, while minimizing makespan (i.e maximizing productivity).

The job times and quality are encoded onto the edges of a MO-GTSP instance. Each job is mapped to a cluster, and the job configurations are the nodes inside a cluster. If a job needs to be executed multiple times, we create the required amount of instances of the corresponding cluster.

### B. *Details of reference algorithm*

We extend the state-of-the art single-objective MA-GTSP to estimate solutions for the multiple objective variant following the weighted-sum method [12]. This method minimizes a positively weighted sum of the objectives:

[1] http://www.cs.nott.ac.uk/~dxk/gtsp.html

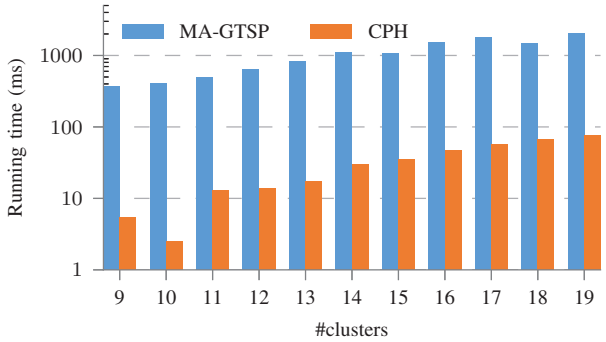



Fig. 4: Running time when a MO-GTSP instance of the given size is extended with one cluster.

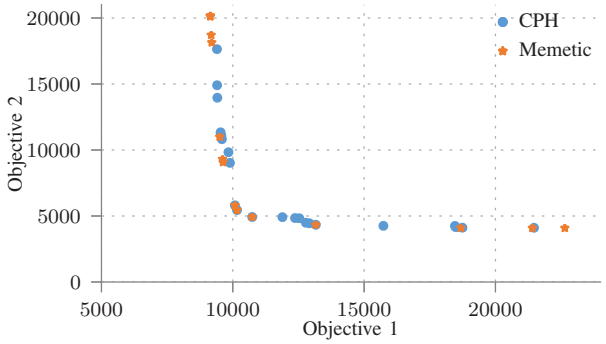



Fig. 5: Example comparison of Pareto fronts generated for the benchmark of 20kroAB100

$\min \sum_{i=1}^{n} (\gamma_i \cdot f_i(x)), \gamma_i > 0$. We can find all convex Pareto points by choosing different objective weights with $\gamma_i \in (0, 1)$. The memetic heuristic of [8] and its implementation is used to solve the single objective instances. The resulting solutions are evaluated to obtain the multi-objective values.

### C. Pareto front comparison

We want to compare the generated Pareto fronts for an indication of the relative quality of the generated solutions. We use the Epsilon $I_\epsilon(A, B)$ and Hyper-volume ratio $\frac{I_h(A)}{I_h(B)}$ indicators [15] to gain insight into the relative quality of solutions. The Epsilon indicator between Pareto front 1 (PF1) and Pareto front 2 (PF2) is the largest multiplication factor of the solution values of PF2 such that PF2 still dominates PF1. The Hyper-volume ratio considers the relative surface (or hyper-volume) of the Pareto front with respect to some point; we take the worst observed values per dimension. For the Epsilon indicator a smaller number indicates relatively better solution quality, and for the Hyper-volume ratio indicator a larger number indicates a relatively better solution quality. The Epsilon and hyper-volume indicators combined gives a better assessment of the relation between the results of two algorithms [15] than the individual indicators.

### D. Offline optimization

The offline evaluation shows how CPH-GTSP performs compared to MA-GTSP. We consider that all information is available at the start of the algorithm. The details of the experiment, including the benchmark files, and the results, can be found at www.es.ele.tue.nl/pareto/mogtsp.

The results show that the wall time required for CPH-GTSP is 5 to 20 times smaller than for MA-GTSP. The runtime of CPH-GTSP is in the order of 100 milliseconds for medium-sized instances. For larger instances, the runtime is in the order of several seconds.

The Hyper-volume ratio indicates that the Pareto fronts generated by CPH-GTSP typically have a slightly lower quality (on average 0.968), and is always in favour of MA-GTSP, i.e. it is always less than 1. The results show, however that MA-GTSP and CPH-GTSP often share several solutions and that CPH-GTSP finds several non-convex points not found by the extension of MA-GTSP (for example see Fig. 5). The Epsilon indicators show that the solutions generated by MA-GTSP are only dominating the CPH-GTSP solutions in 1 out of 26 instances. We observe that CPH-GTSP strengthened with the 2-opt/3-opt and cluster optimization improvement operators yields high-quality results in a short execution time.

### E. Online optimization

We also assess how well CPH-GTSP performs as an online algorithm. To do so, we measure the time the algorithms need to (re-)compute a problem instance extended with a new cluster. The resulting runtime is presented in Fig. 4. The results of the experiments show that CPH-GTSP is significantly faster than MA-GTSP.

## VI. Conclusion

We have shown that the Compositional Pareto-algebraic Heuristic combined with suitable improvement operators yields good results for MO-GTSP. CPH-GTSP is suitable to online computation due to its runtime and compositional nature.

For future work, it is interesting to see if CPH applies to other multi-objective and multiple-choice scheduling and packing problems. These problems are, for example, the multi-objective flow-shop problem, or the multi-objective bin-packing problem. Another approach to solve the MO-GTSP could be to replace the ILS component in SLS with the local search heuristics of MA-GTSP.

## References

[1] G. Reinelt, "TSPLIB A traveling salesman problem library," *ORSA J. Comput.*, 1991.

[2] H. Shojaei *et al.*, "A parameterized compositional multi-dimensional multiple-choice knapsack heuristic for CMP run-time management," in *DAC'09*. ACM, pp. 917–922.

[3] H. Shojaei *et al.*, "A fast and scalable multidimensional multiple-choice knapsack heuristic," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 4, pp. 51:1–51:32, Oct. 2013.

[4] L. Paquete and T. Stützle, "Design and analysis of stochastic local search for the multiobjective traveling salesman problem," *Comput. Oper. Res.*, vol. 36, no. 9, pp. 2619 – 2631, 2009.

[5] T. Stützle and H. H. Hoos, "Analyzing the run-time behaviour of iterated local search for the TSP," in *III Metaheuristics Int. Conf.*, 1999.

[6] V. Pillac *et al.*, "A review of dynamic vehicle routing problems," *Eur. J. Oper. Res.*, vol. 225, no. 1, pp. 1–11, Feb. 2013.

[7] N. Jozefowiez *et al.*, "Multi-objective vehicle routing problems," *Eur. J. Oper. Res.*, vol. 189, no. 2, pp. 293 – 309, 2008.

[8] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem," *Natural Computing*, vol. 9, no. 1, pp. 47–60, 2010.

[9] M. Fischetti *et al.*, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Oper. Res.*, vol. 45, no. 3, pp. 378–394, 1997.

[10] M. Geilen *et al.*, "An algebra of Pareto points," in *Fundamenta Informaticae*. IEEE Computer Society Press, 2005, pp. 88–97.

[11] M. Geilen and T. Basten, "A calculator for Pareto points," in *DATE'07*. IEEE, pp. 1–6.

[12] M. Ehrgott, "A discussion of scalarization techniques for multiple objective integer programming," *OR*, vol. 147, pp. 343–360, 2006.

[13] D. Karapetyan and G. Gutin, "Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem," *Eur. J. of Oper. Res.*, vol. 219, pp. 234–251, 2012.

[14] U. Waqas *et al.*, "A re-entrant flowshop heuristic for online scheduling of the paper path in a large scale printer," in *DATE'15*, pp. 573–578.

[15] E. Zitzler *et al.*, "Quality assessment of pareto set approximations," in *Multiobjective Optimization*. Springer, 2008, pp. 373–404.