TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science

ITV70LT

Priit Järv, 111831IAPMM

A PARALLELIZED METAHEURISTIC FOR GENERALIZATIONS OF THE ORIENTEERING PROBLEM

Master's Thesis

Supervisor: Tanel Tammet
PhD
professor

Tallinn 2015

Declaration

I hereby declare that the presented paper is a result of my own work and that is has not been submitted for defense by anyone else.

. . . .
(date)

. . . .
(signature)

Abstract

We present a parallelization method for algorithms that solve a family of combinatorial optimization problems called the orienteering problem (OP) and its generalizations. This problem arises in the fields of logistics and automated planning and consists of finding a most profitable route between fixed locations, under time limit. An example of such a problem is automated creation of tourist trip plans, where the goal is to create a tour between places that are most interesting for the tourist, given that the tourist has a certain amount of time available.

We use an existing non-parallel algorithm that belongs to a group of generic optimization techniques called metaheuristics. We develop a novel parallelization method that benefits from cooperative behaviour. For comparison, we also examine two other parallelization approaches. We implement a non-cooperative parallel version of the same algorithm. We also implement a known parallel algorithm for similar metaheuristics called greedy randomized adaptive search procedure with path relinking (GRASP-PR).

We compare our parallelization technique to these alternative versions to show that the cooperative approach contributes to the solution quality and speed and that our novel approach is competitive with a previously published approach. For comparison, we perform tests on five benchmark datasets from the literature. The results are also compared to non-parallel state of the art algorithms that have produced significant results on these benchmarks.

We show that our approach exhibits the useful property of reaching good quality suboptimal solutions early in the search, with predictable probability. We additionally show that the technique scales well with up to 128 parallel worker processes. The quality of the solutions varies depending on the benchmark, in some problem sets our results exceed those of specialised algorithms, in other cases they remain inferior.

Annotatsioon

Käesolevas töös esitletakse parallelset meetodit lahendamaks kombinatoorikaülesandeid, mis kuuluvad orienteerumisülesande ja selle laienduste hulka. Orienteerumisülesanne kuulub logistika ja automatiseeritud planeerimise ülesannete valdkonda. Selle eesmärgiks on piiratud aja jooksul leida võimalikult kasulik marsruut ette määratud objektide vahel. Näide orienteerumisülesandest on automaatselt turisti reisiplaani koostamine, kus turisti päevaplaani peaks koostama selliselt, et turist saaks külastada talle kõige rohkem meeldivaid vaatamisväärsusi.

Aluseks on võetud olemasolev mitteparallelne algoritm, mis kuulub üldistatud lahendusmeetodite perekonda nimega metaheuristikud. Töös kirjeldatakse uut paralleliseerimise tehnikat mis toetub koostööle paralleelsete komponentide vahel. Võrdluse eesmärgil on töös kasutatud kahte täiendavat paralleelset algoritmi - variant samast algoritmist, kus koostööd ei kasutada ning varasemalt erialases kirjanduses tuntud algoritm mida on kasutatud sarnaste metaheuristikute puhul.

Võrdlus teostatakse, rakendades kõiki kolme paralleelset algoritmi viiele erialases kirjanduses avaldatud ülesannete paketile. Selle eesmärgiks on näidata, et koostöö kasutamine annab algoritmile lisaväärtust ning lähenemine on ühtlasi konkurentsivõimeline juba teadaoleva meetodiga. Saadud tulemusi võrreldakse ka parimate avaldatud tulemustega erialastest artiklitest.

Töös näidatakse, et valitud lähenemisel on kasulik omadus jõuda hea kvaliteediga tulemusteni otsingu varajases faasis, sõltumatult ajast mis kulub lõpliku tulemuse leidmisele. Ühtlasi näitavad tulemused, et esitletud koostööl põhinev lahendus skaleerub hästi kuni 128 paralleelse protsessini. Tulemuste kvaliteet varieerub võrreldes spetsiifilistele ülesannetele kohaldatud algoritmidega, ületades neid kohati, aga mitte konsistentselt.

Contents

Chapter 1

Introduction

This research presents a parallel algorithm that is designed to solve the orienteering problem (OP) and its extensions, with the focus on those related to automatically creating tourist trip itineraries. We build the parallelization on the foundation of an existing non-parallel algorithm that has previously exhibited good results on problems belonging to the same class. We aim to show that our parallel method benefits from cooperative behaviour and is scalable in parallel environments.

The OP is a combinatorial optimization problem that mathematically models various logistics and planning tasks. The name originates from the sport of orienteering where the participants, equipped with a topographical map and compass, visit the control points placed on terrain. In a variant called score orienteering, the control points have scores assigned based on the difficulty of reaching or locating them. The participants must plan the most profitable path between control points that they are able to complete within a given time budget. (Tsiligirides 1984)

In the OP, the locations to be visited can be modeled as the vertices of a graph. The vertices are associated with rewards and the edges with costs. The goal is to maximize the reward collected by visiting the vertices without exceeding a given limit of accumulated cost. The solution is a subset of vertices. The OP is different from the travelling salesman problem (TSP) where the goal is to visit all vertices and to minimize the accumulated cost. In TSP there is no limit of the accumulated cost. However, the problems are closely related in that the optimum solution requires finding the shortest path between subsets of vertices.

The OP is being applied in various areas of automated planning. Examples include routing a fleet of service vehicles that need to visit customers in various locations, planning tourist itineraries (Vansteenwegen et al. 2011) and path planning for law enforcement surveillance vehicles (Pietz and Royset 2013). Our interest in the OP stems from tourist trip recommender system research. The problem of recommending an itinerary for a trip or a tour is called the tourist trip design problem (TTDP) (Vansteenwegen and Oudheusden 2007). The trip must fit within a given time budget and include the points of interest (POIs) that are most attractive to the tourist.

The OP and its extensions have been the most common model for solving the TTDP. In the context of the TTDP, the vertices represent POIs and the edges usually represent the cost of movement. In tourist trip recommender systems, various additional parameters, such as the opening hours, multi-day visits, budget constraints and dependency on transport networks need to be observed. The basic OP model is commonly extended to include these features, leading to various generalizations of the OP. (Gavalas et al. 2014)

Metaheuristics are an attractive method of solving computationally demanding problems. In common with the more ubiquitous concept of heuristics, they offer an approximating approach that does not guarantee finding an optimal solution. Instead, the goal is to find a solution that is ”good enough”, ”fast enough”. However, metaheuristics are viewed as a template or set of guidelines rather than a specific algorithm. Their appeal is the ability to offer solutions to a wide range of problems with predictable results. For example, Aiex et al. (2002) show experimentally that the probability of finding a solution with predetermined quality using the greedy randomized adaptive search procedure (GRASP) metaheuristic fits theoretical probability distribution.

1.1 Related work

Modeling the TTDP is non-trivial, as there are several characteristic features that different authors consider to be essential in producing quality solutions:

- multi-constraint optimization (Vansteenwegen and Oudheusden 2007; Gavalas et al. 2014); including multiple time windows (Gavalas et al. 2014)
- multi-criteria optimization (Vansteenwegen and Oudheusden 2007; Schilde et al. 2009; Rodríguez et al. 2012)
- time-dependence of costs (Fomin and Lingas 2002; Verbeeck et al. 2014) and rewards (Hasuike et al. 2013; Erdogan and Laporte 2013; Yu et al. 2014)
- inter-dependence of the POIs (Vansteenwegen and Oudheusden 2007; Gionis et al. 2014).

We are not aware of any OP generalizations that would encompass all of these features. In the following we focus on publications that solve problems that contain a subset of these features and represent the best-performing algorithms. The reader is referred to Vansteenwegen et al. (2011) and Gavalas et al. (2014) for a comprehensive overview of the OP and TTDP literature.

Customizing the team orienteering problem (TOP) (Chao et al. 1996a) for tourist trip recommendation has produced a family of OP variants that include constraints. The team orienteering problem with time windows (TOPTW) (Vansteenwegen et al. 2009) is the most actively researched of these. Cura (2014) provides a summary of recent benchmark results. In this variant, vertices have legal time windows during which they may be visited; this represents the opening or accessible hours of POIs. The multi-constrained team orienteering problem with (multiple) time windows (MCTOP(M)TW) adds budget and max-n type constraints (García et al. 2013; Souffriau et al. 2013). Currently, the GRASP-ILS hybrid (Souffriau et al. 2013) has the best average gap, but the tabu search of (Sylejmani et al. 2012) holds the best solution in 70 of the test instances.

Several papers treat the TTDP as a multi-objective optimization problem (Schilde et al. 2009; Rodríguez et al. 2012; Hasuike et al. 2014). Inevitably, without context-specific information about the user preferences towards the criteria, a multi-objective problem can only be optimized to the Pareto front of solutions (Marler and Arora 2004; Schilde et al. 2009). This can leave a large number of alternative recommendations and can be challenging to present in a user-friendly manner (Adomavicius et al. 2011). Rodríguez et al. solve this with an interactive step following the search.

In this paper, we only consider the applications where the goal is to non-interactively find a single best solution. If the model includes multiple criteria, then these may be aggregated into a single nonlinear objective function. For example, Geem et al. (2005); Wang et al. (2008); Silberholz and Golden (2010) study an instance of the generalized orienteering problem (GOP) where each vertex of the network has a set of attributes. The objective function aggregates the objective values computed by attribute as a weighted sum (the weights vector represents the prior preference). The two-parameter iterative algorithm (2-PIA) of Silberholz and Golden (2010) dominates the other published algorithms on the benchmark dataset that consists of 27 POIs.

Dynamic models try to realistically convey the changes in urban environments (time dependent costs) and varied user satisfaction depending on the time and duration of visiting a vertex (time dependent rewards). The time dependent orienteering problem (TDOP) (Fomin and Lingas 2002) is a formulation of the former. Currently only Verbeeck et al. (2014) include a benchmark for experimental evaluation.

OP variants with time dependent rewards have emerged recently. Hasuike et al. (2013) present a model where the reward collected at a vertex is a function of time. This is extended to include multiple criteria, classes of POIs (from ”must visit” to ”don’t care”) and resource accumulation in (Hasuike et al. 2014). Erdogan and Laporte formulate the orienteering problem with variable profits (OPVP), where the reward is a function of number of passes at a vertex (OPVP1) or a function of the time spent at the vertex (OPVP2) (Erdogan and Laporte 2013). The reward-maximising variant of optimal tourist problem (OTP) (Yu et al. 2014) is similar to OPVP2. Their solution method accommodates arbitrary reward collection functions by finding piecewise linear approximations. Reproducible gap results to benchmark instances are only reported in (Erdogan and Laporte 2013).

There are few papers that consider the inter-dependence of POIs. Gionis et al. (2014) propose two formulations that classify the POIs into types and introduce an ordering constraint of types. Depending on the parameters, the GOP objective function as formulated in (Silberholz and Golden 2010) also makes the reward of a POI dependent on the other POIs present in the solution.

To the best of our knowledge, no parallelized TTDP solvers have been published to date (Gavalas et al. 2014). Mocholí et al. (2005) propose a parallel grid ant colony algorithm for the OP. They report super-linear speedup in the larger instances of 10000 vertices with up to 32 computing nodes. Catalá et al. (2007) improve on the solution quality obtained using graphical processing units (GPUs), however their scaling behaviour is poor in comparison. Parallel approaches to related routing problems are numerous (Alba et al. 2013), with the recent focus on GPU computing (Schulz et al. 2013).

An overview and extensive bibliography of the greedy randomized adaptive search procedure (GRASP) can be found in (Resende and Ribeiro 2010). GRASP parallelization strategies are divided into independent and cooperative approaches. The independent strategy involves either trivially partitioning GRASP iterations between processors, starting each with an independent random seed or partitioning the search space. Threads only share the global best solution (Resende and Ribeiro 2005). An early example of the partitioning method can be found in (Feo et al. 1994). The cooperative strategy implies that the threads collaborate in order to speed up local progress. This has been achieved through the path relinking technique by sharing elite solutions (Resende and Ribeiro 2005). The pool of elite solutions can either be local (Aiex and Resende 2005) or global (Ribeiro and Rosseti 2007).

1.2 Objectives and organization

The primary objective of the research presented here is to introduce a parallel solution method for the TTDP. We treat the task of constructing a trip automatically as the problem of solving OP generalizations on a graph. To solve OP variants that are motivated by the TTDP, we rely on established methods of parallel metaheuristics. We adapt an existing sequential heuristic to a parallel method in a novel way.

This paper presents a cooperative parallelized GRASP-ILS hybrid for generalizations of the OP. The algorithm is a generic template that can handle non-linear or even non-monotonic objective functions. We show that it can be adapted for several of the generalizations discussed above. We present the performance evaluation of our parallelization using previously published benchmarks.

In addition to our novel parallelization techniques, we develop two parallelization approaches to serve as a point of comparison. First, we use a non-cooperative version of the same GRASP-ILS hybrid to determine the effect of cooperation on the solution quality and speed. Second, we present a similar generic template based on an existing parallel metaheuristic GRASP-PR, to be able to compare against a previously published parallelization method.

For each OP variant examined, we develop problem-specific components that can be incorporated into the generic algorithms. The problem-specific algorithms are then implemented and tested in a cluster environment.

The rest of the paper is organized as follows. In Chapter 2 we present the sequential algorithms that form the foundation of our parallelized versions. We then proceed to present the algorithms implemented for comparison and finally our novel parallelization technique in Section 2.3. Chapter 3 presents the results of the experiments performed on a high performance computing cluster using existing benchmarks from the literature. For each benchmark, we also compare the parallel techniques with the state of the art. In Chapter 4 we summarize the findings and discuss further perspectives in applying and enhancing the solution methods. Detailed benchmark results are included in Appendix A.

Chapter 2

Hybrid metaheuristic template

We initially present the algorithms as a generic template. For each problem, the specific features are then added as required. These problem-specific functions are described in Chapter 3.

We begin by describing the sequential metaheuristics that the parallelization techniques are built on. The first algorithm (Section 2.1) is pure GRASP enhanced with path relinking (GRASP-PR), which is a common technique used in many combinatorial search problems (Resende and Ribeiro 2010). The second algorithm (Section 2.2) is a hybridization of GRASP and Iterated Local Search (ILS).

We then describe three parallelization methods: a cooperative parallel GRASP-PR and an independent-walk strategy version of the GRASP-ILS hybrid are implemented for comparison. Finally we present the cooperative version of the GRASP-ILS hybrid that is the main subject of study in this paper (Section 2.3).

2.1 GRASP with path relinking

The GRASP is a family of search algorithms that are based on the idea that introducing some randomness in greedy search provides better coverage of the search space and helps escaping local optima. The basic template consists of the randomized greedy construction phase that builds a feasible solution and a local search phase that then finds the local optimum in the neighbourhood of the solution. The parameter α controls the randomness in the construction phase. In the construction phase, items are added to the solution iteratively. Each consecutive item is picked randomly from a subset of the items, called restricted candidate list (RCL). The size of the RCL is controlled by α. If $\alpha = 0$ then only the item that appears to be the best remains in the RCL and the construction degenerates to greedy search. If $\alpha = 1$ then the next insertion is made randomly. (Resende and Ribeiro 2010)

We adapt the canonical GRASP template to the OP as follows (Algorithm 1). In line 1, the pool of elite solutions is initialized. The pool holds the current best known solution as well as a number of other good solutions. Each iteration of the loop body between lines 2-17 produces a new solution. If the solution is good enough, it is added to the elite pool. The loop is repeated a fixed number of times. After the loop terminates, the best solution in the elite pool is the best solution that the search found.

Algorithm 1 GRASP with path relinking

1: $ElitePool \leftarrow \emptyset$
2: **for** $i \leftarrow 1, MaxIterations$ **do**
3:     $\alpha \leftarrow$ RandomUniform$(0, 1)$
4:     Determine the heuristic value for each candidate
5:     $Solution \leftarrow \emptyset$
6:     **while** $Solution$ is not full **do**
7:         $RCL \leftarrow$ MakeRCL(α)
8:         Pick a random candidate from RCL and insert into $Solution$
9:         Recalculate the heuristic value for each candidate
10:     **end while**
11:     $Solution \leftarrow$ LocalSearch$(Solution)$
12:     $Guiding \leftarrow$ FindGuidingSolution$(Solution, ElitePool)$
13:     $Solution \leftarrow$ PathRelinking$(Solution, Guiding)$
14:     **if** $Solution$ is elite **then**
15:         Update $ElitePool$ with $Solution$
16:     **end if**
17: **end for**

Lines 3-5 initialize the search for the current iteration. We fix the parameter α in the interval $[0, 1]$. There are several published techniques for finding a suitable value for α (Resende and Ribeiro 2010). Usually the best value depends on the specific problem instance and on the heuristic function used to evaluate the candidates. Here α is chosen randomly from the uniform distribution. This ensures that each candidate has a non-zero probability of being included in the construction phase.

Before the construction starts, the list of candidate insertion moves is prepared. An insertion move is the pair (v, i) of vertex v and the insertion point i. If the insertion move is executed, v is placed in the solution after the vertex at position $i - 1$ and the remaining vertices $i, i + 1, i + 2, \ldots$ are shifted by one place.

If the set of vertices is V and the insertion points $i = (1, 2, 3, ...)$ then the candidate list $C = \{(v, i_{max}) \mid v \in V, i_{max} = \arg\max_i h(v, i)\}$. The heuristic score $h(v, i)$ allows differentiating the possible insertion positions to optimize the path length. In the generalized model, the heuristic value $h(v, i)$ depends on the insertion point as well as the contents of the partial solution. The actual heuristic function is problem-specific. Further details will be given in Sections 3.1-3.5.

In case of models with time windows, rearranging the path in the complete solution, for example by using λ-opt search moves, is less effective (Potvin and Robillard 1995). Because of that we only consider the locally best insertion for each vertex. Additionally, we only consider moves that result in feasible solutions. In case the heuristic function is expensive, this reduces the computational effort, as there is no need to compute the heuristic value for non-feasible candidates.

We note that Souffriau et al. (2008) have described a similar approach for the team orienteering problem, except that they include all feasible insertion moves in the candidate list.

Lines 6-10 constitute the construction phase that builds the solution by choosing moves from a list of candidate insertion moves. The construction phase terminates when there are no more candidate moves that can be applied to the current solution without violating any constraints.

The RCL is formed by taking a subset of the candidate list (line 7). Let h_{min} and h_{max} be the lowest and highest heuristic values of the candidate moves. We determine the threshold value $h_t = h_{min} + (1 - \alpha)\,(h_{max} - h_{min})$ and the $C_{RCL} =$

$\{(v, i) \in C \mid h(v, i) \geq h_t$. A random element (v, i) of the RCL is selected and vertex v is inserted into the current solution at position i (line 8).

In some generalizations of the OP the objective function may be dynamically changing during the construction of the solution. During the greedy construction phase, the heuristic value[1] of the candidate vertices is re-evaluated after each insertion into the solution (line 9). Thanks to this ”adaptive” aspect, the metaheuristic is well suited for optimizing arbitrary objective functions.

In line 11, the solution is optimized using local search moves. In general, this involves re-ordering, inserting and removing vertices in the solution, until a local optimum is reached. The local search is problem-specific, as the effectiveness of search techniques varies depending on how constrained the problem is. It can also be completely omitted. We describe the appropriate local search for each problem in Sections 3.1-3.5.

The path relinking phase (lines 12-13) then attempts to enhance the current solution further. One elite solution is selected as the guiding solution. We choose the guiding solution randomly from the elite pool. A solution can be transformed into another solution by applying successive search moves, such as insertions and removals. This forms a path in the search space between the solutions. The path relinking procedure follows this path and attempts to find intermediate solutions that are better than the current solution. The expectation is that making the current solution more similar to an elite solution improves it, while making an elite solution more similar to an arbitrary solution diversifies the solutions the algorithm visits.

Lines 14-16 update the pool of elite solutions. A solution is considered eligible for the elite pool, if:

- the pool is not yet full;
- it has a higher score than any of the current elite pool members, or
- if it has a higher score that at least one of the elite pool members and is sufficiently different from them.

[1] Resende and Ribeiro use the term ”greedy evaluation”.

We define the measure of similarity between two solutions A and B to be $\frac{2|A \cup B|}{|A|+|B|}$ (Souffriau 2010).

Algorithm 2 gives the path relinking procedure in detail. We use the mixed path relinking technique (Resende and Ribeiro 2010) where the path between the current and guiding solution is built from both ends simultaneously.

Lines 2-3 prepare the search procedure. $BestSolution$ keeps track of the locally best solution found on the search path. The set $Difference$ consists of the vertices in the guiding solution that are not present in the current solution. From this point, the sets $Solution$ and $Guiding$ represent the ends of the paths originating from the current solution and the guiding solution. The search will repeat lines 4-16 until one of $Solution$ and $Guiding$ becomes a subset of the other. At this point the solutions contain the same vertices, or no further progress will be made by making them more similar, as this can only be accomplished by removing vertices.

Lines 5-6 greedily select the vertex with the best heuristic value from the set $Difference$. The heuristic value computed with the same function as in the construction phase, except that the insertion may cause the solution to become infeasible.

Lines 7-10 repair and optimize the working solution. The vertex with the lowest heuristic value is removed, provided that $v \notin Solution \cap Guiding$. The additional condition is needed to prevent cycles in the path that would result from removing vertices that earlier steps have added. If there are no such vertices, the repair step is allowed to fail in which case the search terminates early and the current $BestSolution$ is returned. Next, local search is applied to tighten the path. This is repeated until the working solution becomes feasible.

In lines 11-13 the working solution is compared to the best known solution along the path. $BestSolution$ is updated, if necessary. Lines 14 and 15 then switch to the other end of the connecting path by swapping $Solution$ and $Guiding$ and computing the new difference. If $Difference$ becomes an empty set, the termination condition is met (no further progress can be made) and the best solution found along the path is returned.

Algorithm 2 Path relinking procedure

procedure PATHRELINKING($Solution, Guiding$)
    $BestSolution \leftarrow Solution$
    $Difference \leftarrow Guiding \setminus Solution$
    while $Difference \neq \emptyset$ **do**
        Compute the heuristic value for each item in $Difference$
        Insert the item with best value into $Solution$
        while $Solution$ is not feasible **do**
            Remove the item with worst heuristic value from $Solution$
            $Solution \leftarrow$ LOCALSEARCH($Solution$)
        end while
        if $Solution$ is better than $BestSolution$ **then**
            $BestSolution \leftarrow Solution$
        end if
        Swap $Guiding$ and $Solution$
        $Difference \leftarrow Guiding \setminus Solution$
    end while
    return $BestSolution$
end procedure

2.2 Hybrid GRASP-ILS metaheuristic

There are several approaches to hybridizing GRASP with iterated local search. Ribeiro and Urrutia (2007) apply ILS as the local search after the greedy construction phase (ILS inside GRASP). In solving MCTOP(M)TW, Souffriau et al. (2013) use greedy random construction to build the solution in each iteration (greedy random construction inside ILS). We have used the latter approach with some modifications (Algorithm 3).

The algorithm shares the general structure with pure GRASP. The main difference is that each random construction step is seeded with a partial solution instead of an empty one. The seed solution is derived by perturbing the solution from previous iteration. Some items, with a bias towards the one that has lowest heuristic value, are removed. They may also be temporarily eliminated from being used in the construction.

The partial solutions carry over knowledge from previous iterations. The elimination bias is helpful in diversifying the solutions - otherwise the items removed in the perturbation step may re-enter the solution immediately, in the extreme case even reproducing a solution identical to the previous iteration.

Algorithm 3 GRASP-ILS hybrid

1: $BestSolution \leftarrow \emptyset$
2: $Solution \leftarrow \emptyset$
3: **for** $i \leftarrow 1, MaxIterations$ **do**
4:     $\alpha \leftarrow$ FINDALPHA()
5:     Determine the heuristic value for each candidate
6:     **while** $Solution$ is not full **do**
7:         $RCL \leftarrow$ MAKERCL(α)
8:         Pick a random candidate from RCL and insert into $Solution$
9:         Recalculate the heuristic value for each candidate
10:     **end while**
11:     $Solution \leftarrow$ LOCALSEARCH($Solution$)
12:     **if** $Solution$ is better than $BestSolution$ **then**
13:         $BestSolution \leftarrow Solution$
14:     **end if**
15:     $Solution \leftarrow$ PERTURB($Solution, \beta$)
16: **end for**

Lines 1-2 initialize the known best solution and the current working solution. The loop body between lines 3-16 then attempts to improve the working solution by repetitively adding items, then removing a random subset of them. After the loop terminates, $BestSolution$ holds the best solution that the search found.

Each iteration begins with determining α (line 4) using the ”Reactive GRASP” approach (Prais and Ribeiro 2000). At each iteration, α is chosen randomly from the set $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. These values are predetermined. The probability that α_i is chosen is initially set to $p_i = \frac{1}{m}$. We keep track of the average solution score $\bar{S}_i$ resulting from using α_i. Periodically, the probability distribution is updated as follows. Let S^* be the current value of $BestSolution$. For each α_i, we then compute the ”quality” $q_i = \left(\frac{\bar{S}_i}{S^*}\right)^\delta$ and the probability $p_i = \frac{q_i}{\sum_{j=1}^{m} q_j}$. The values of α that result in higher solution scores on the average, consequently receive higher probabilities of being chosen again. The parameter δ can be used to accelerate this process. Table 2.1 gives the values of parameters used in the experiments.

Lines 5-10 constitute the construction phase that is identical to the GRASP with path relinking (Section 2.1). A candidate list of insertion moves (v, i) is formed by assigning each vertex v a heuristic score $h(v, i)$ that corresponds to the insertion in the optimal position i. The RCL is then constructed by taking candidates that have heuristic values over a certain threshold that depends on the current α. A random move from the RCL is performed, after which the heuristic scores for all vertices and insertions are recalculated (this is required because the heuristic score is dependent on the current solution).

Similarly, line 11 performs a hill climbing local search which is problem dependent and is specified in Sections 3.1-3.5. This completes the solution that the current iteration has built and lines 12-14 update the $BestSolution$, if necessary.

Line 15 performs the perturbation of the current solution. This is the main departure from the canonical GRASP template, as the resulting partial solution will then become the initial state of the construction phase of the next iteration. The parameter β determines the percentage of solution elements that are removed at the end of the iteration. Let m be the current number of elements that are eligible for removal. Some problems instances specify mandatory elements, such as start and end vertices, which should not be removed. Let S_i the score after removing the vertex i from the solution. Let k be the index of a vertex such that $\forall i \neq k S_k \geq S_i$. The probability of removing vertex $i \neq k$ is then $p_i = \frac{1}{m+1}$. The perturbation phase is biased towards removing the vertex k, with probability $p_k = \frac{2}{m+1}$. Once an item is removed, it is additionally eliminated from the construction phase for a small number of iterations. This is similar to the "perturb by elimination"[2] strategy (Resende and Ribeiro 2010). Table 2.1 gives the empirically chosen parameters used in the experiments reported here.

Short elimination duration is preferable for relatively small problem instances. Otherwise a significant portion of the items are simultaneously eliminated and the search progresses slower. By relying on empiric evidence, we've omitted elimination completely in the experiments. The parameter β influences how close

[2]The word "perturb" has a different meaning in this context, referring to changes in the heuristic value

Table 2.1: Parameters of the GRASP-ILS hybrid used in the experiments. δ and A apply to Reactive GRASP technique, the rest of the parameters are used in the perturbation phase

Parameter	*Value*
δ	2
A	$\{0.1, 0.2, \ldots, 0.9\}$ $(m = 10)$
β	random with biased distribution
Elimination duration	none (items are immediately available again)

consecutive solutions are to each other in the search space. We choose β randomly from a set biased towards smaller perturbation. This allows the search to remain in the same neighbourhood for longer. It is also possible to borrow the idea of the simulated annealing search and start with a large value of β, decreasing it gradually. This would leads to perturbations decreasing with time.

2.3 Parallelization techniques

Pure GRASP is well suited for parallelization. Because of the independence of the iterations the work may be partitioned between the processors with the expectation of linear speedup (Aiex et al. 2002). While introducing memory into the search makes its results less predictable theoretically (Resende and Ribeiro 2010), simple partitioning has practical benefits when applied to the memory-enhanced search as well (for example, as shown experimentally by Aiex and Resende (2005)).

We implement an independent parallel strategy (Algorithm 4) to serve both as a baseline comparison and as a viable approach in case the communication between processes is expensive. The messaging is limited to maintaining the global best solution and to detect process termination. The process is locally aware of the lower bound on the global best solution. Sending a message is only necessary when the last known global best is exceeded. At this point the lower bound is also updated.

There is a separate monitor process that communicates with each worker process individually. The monitor is responsible for storing the global best solution

Algorithm 4 Independent parallel GRASP-ILS hybrid

1: $KnownBestSolution \leftarrow \emptyset$
2: $Solution \leftarrow \emptyset$
3: **for** $i \leftarrow 1, MaxIterations$ **do**
4:     ... Build $Solution$ ...
5:     **if** $Solution$ is better than $KnownBestSolution$ **then**
6:         SEND($Solution$)
7:         $KnownBestSolution \leftarrow$ RECEIVE($GlobalBestSolution$)
8:     **end if**
9:     ... Finish the iteration ...
10: **end for**
11: SEND($Termination$)

and reporting the search termination once all the processes have finished. Here and in the following parallel algorithms, we define the primitive SEND() to be a procedure that sends the specified message synchronously. The primitive RECEIVE() receives the specified message from another process synchronously.

For brevity, we've omitted parts of the parallelized algorithms that do not differ from the sequential versions. At line 1, the local best solution is initialized. Lines 2-4 correspond to the construction phase and local search of the sequential GRASP-ILS algorithm (lines 2-11 in Algorithm 3). At line 5 we compare the current working solution to the local best solution. If it is not better than the local best solution, then it also cannot be better than the global best solution and we may skip communicating with the monitor process. Otherwise, we send the solution to the monitor process as a new potential best solution. Regardless of whether this results the global best solution being updated, the monitor process responds by sending the global best which is stored locally in $KnownBestSolution$.

Lines 9-10 correspond to lines 15-16 in Algorithm 3. The solution is perturbed by eliminating elements randomly and the iteration ends. After the loop terminates, $KnownBestSolution$ is discarded. The worker process sends a termination message to the monitor. Once the monitor has received termination messages from all workers, it will also terminate. The global best solution stored by the monitor process is the end result of the parallelized search.

We also present cooperative parallelizations for both sequential metaheuristics

(Sections 2.1 and 2.2). The GRASP with path relinking is parallelized by sharing the elite pool between processes. The collaboration is achieved by distributing elite solutions found by one worker to all other workers. To simplify the algorithm, we do not use local elite pools. Instead, we implement a monitor process that stores the global elite pool. When the search needs to interact with the elite pool, it will communicate with the monitor process.

Algorithm 5 Cooperative GRASP with path relinking

1: **for** $i \leftarrow 1, MaxIterations$ **do**
2:     ... Build $Solution$ ...
3:     SEND($Solution$)
4:     $Guiding \leftarrow$ RECEIVE($GuidingSolution$)
5:     $Solution \leftarrow$ PATHRELINKING($Solution, Guiding$)
6:     SEND($Solution$)
7: **end for**
8: SEND($Termination$)

Algorithm 5 represents the worker process. The algorithm does not require any state to be maintained between iterations locally. The search begins immediately with the GRASP main loop (line 1). Each iteration then proceeds to build the solution (line 2) as follows. α is determined randomly and the working solution is initialized to empty. The heuristic value for each vertex is then computed, RCL constructed and one insertion move from the RCL performed. This is repeated until no more vertices fit into the solution, then the solution is post-optimized using local hill climbing search (see Algorithm 1, lines 3-11).

At line 3 the current solution is sent to the monitor process, requesting a suitable guiding solution. Lines 4-5 receive the guiding solution and call the path relinking procedure (Algorithm 2) that examines the search space between the current and guiding solution. The best solution found along that trajectory is sent to the monitor process (line 6). The worker process does not require a response to this message. Again, the parallelized search ends by sending a termination message (line 8).

The monitor process (Algorithm 6) is responsible for maintaining the memory of the search. At line 2, the pool of elite solutions is initialized to empty. The process then loops, waiting for messages from workers (lines 3-17) until each worker has sent a termination message. The process ends by returning the best solution in the elite pool (line 18).

Algorithm 6 The monitor process for maintaining the elite pool

```
1: procedure PoolMonitor
2:     ElitePool ← ∅
3:     while Any workers running do
4:         Receive(Message)
5:         if Message is Termination then
6:             Record that the sender has terminated
7:         else if Message is Solution then
8:             if this is a guiding solution request then
9:                 Find Guiding solution from the pool
10:                Send(Guiding)
11:            else
12:                if Solution is elite then
13:                    Update ElitePool with Solution
14:                end if
15:            end if
16:        end if
17:    end while
18:    return the best solution in ElitePool
19: end procedure
```

At line 4 the RECEIVE() primitive is called. It blocks until a message arrives from any of the workers. The type of the message determines the response from the monitor process. If the message is a termination notification then the sender is marked as no longer running (line 6). If this was the last worker running, the test on line 3 fails and the loop ends.

If the message contains a solution, then it is either a request for a guiding solution or a request to add the message to the elite pool. In the first case, a random solution is taken from the elite pool and sent to the worker process (lines 9-10). In the second case, same criteria as in the sequential search are applied to determine whether the solution is elite (line 12). If the pool is not yet full, the solution is always added. A globally best solution is also always added. Otherwise, the solution is added if it is better than at least one solution in the pool and its similarity measure to at least one solution is lower than a given threshold. If the elite pool was full, we remove the solution, that is most similar to the added solution among those that have lower score than the added solution (line 13).

The GRASP-ILS hybrid requires a different approach to collaboration. We observe that the method of perturbations creates a connected trajectory through the search space. Similarly to the path relinking technique, we then take the assumption that it is beneficial to explore the neighbourhoods of known good solutions.

The search (Algorithm 7) starts with each worker exploring a different trajectory. Whenever a worker discovers a new global best solution, the rest of the workers copy that solution and proceed to search its neighbourhood by introducing their own perturbations. The trajectories then diverge again, as each worker has an independent randomness source. The speed at which the divergence occurs depends on the distribution of β. We name this technique *trajectory rejoining*.

Algorithm 7 Cooperative parallel GRASP-ILS hybrid

1: $KnownBestSolution \leftarrow \emptyset$
2: $Solution \leftarrow \emptyset$
3: **for** $i \leftarrow 1, MaxIterations$ **do**
4:     . . . Build $Solution$ . . .
5:     **if** $Solution$ is better than $KnownBestSolution$ **then**
6:         SEND($Solution$)
7:     **else**
8:         SEND($PollForBest$)
9:     **end if**
10:     $NewBestSolution \leftarrow$ RECEIVE($GlobalBestSolution$)
11:     **if** $NewBestSolution$ is better than $KnownBestSolution$ **then**
12:         $KnownBestSolution \leftarrow NewBestSolution$
13:         $Solution \leftarrow NewBestSolution$ ▷ trajectory rejoin
14:     **end if**
15:     $Solution \leftarrow$ PERTURB($Solution, \beta$)   ▷ trajectory diverges
16: **end for**
17: SEND($Termination$)

Lines 1-2 initialize the current working solution and the known best solution to empty. The score of the known best solution is a lower bound on the globally best score. Lines 3-16 are repeated a fixed number of times, each time the solution is constructed by adding vertices and then perturbed by removing vertices randomly. After the loop ends, the termination message is sent to the monitor process (line 17).

Line 4 represents building the solution as in lines 4-11 of Algorithm 3. α for the current iteration is determined using the Reactive GRASP technique. Then the solution is filled by performing insertion moves from the RCL until there are no legal candidate moves left. After each insertion the heuristic value of insertions is re-evaluated and RCL rebuilt.

Lines 5-10 are responsible for discovering new best solutions found by other workers. If the search has found a solution that is potentially globally best it is sent to the monitor process (line 6). Otherwise the worker sends a poll to request the current global best solution (line 8). In both cases the monitor process responds by sending the globally best solution that is received by the worker synchronously (line 10).

Lines 11-14 perform the join with the globally best trajectory. If $NewBestSolution$ is better than the local $KnownBestSolution$ then this indicates either that this worker or another worker has found a new globally best solution. Line 12 updates the locally known best. If the new global best was found by another worker then line 13 makes the worker jump to the same point in the search space.

Line 15 is the perturbation phase that is identical to the sequential version of the GRASP-ILS hybrid. A random number of vertices are removed from the solution. They are not used in subsequent construction phases for a fixed number of iterations, to diversify the search.

The monitor process for distributing the trajectory information to the workers is nearly identical to the monitor process for the independent parallel GRASP-ILS. It only needs to additionaly respond to the polls for the current global best solution. The workers have sufficient information to detect when the global best has changed, since by definition it must be better than the locally known best solution.

Chapter 3

Experimental results

We performed experiments on the following problems:

- orienteering problem (OP) (Section 3.1);
- generalized orienteering problem (GOP) as formalized, among others, by Wang et al. (2008) (Section 3.2);
- team orienteering problem with time windows (TOPTW) (Section 3.3);
- multi-constrained team orienteering problem with multiple time windows (MCTOPMTW) (Section 3.4);
- time dependent orienteering problem (TDOP) (Section 3.5).

By covering these problems, the evaluation includes the basic unconstrained model with Euclidean costs and linear objective function as well as constrained models and a model with nonlinear cost function. The generic metaheuristics were adapted to specific problems. We solved problem instances from published benchmark datasets for which known optimal or best known results are available.

The test programs were written in Python in object-oriented style, to facilitate implementing several closely related algorithms and problem models. The interprocess communication was implemented using MPI, provided by the native `mpi4py` module. To reduce the computation time required, most time-consuming parts of the algorithms (building the candidate list and the local search) were rewritten in the Cython language, translated to C and compiled into binary modules.

The tests were executed on a HPC cluster with dual Intel Xeon E5-2630L processors (12 cores and 24 threads total) per node. The nodes were connected with Infiniband network.

The bulk of the test runs was performed node-locally with 23 parallel workers. Each test run consisted of a fixed number of iterations assigned to each worker. These tests produced the solution quality (gap) and average solution time results reported in this chapter.

Additional tests were performed with an increased number of iterations to obtain probability distribution of time to fixed suboptimal solution and varied number of workers for the scalability results.

3.1 Orienteering Problem

Adapting our algorithms to any specific problem requires at least two steps: implementing a heuristic function and implementing the generation of feasible insert moves in the construction step. Additionally, local search should be implemented in cases where it is appropriate.

Let x_{ij} be 1 when the edge from vertex i to j is included in the solution and 0 otherwise. p_j is the reward associated with vertex j. The objective function of the OP can then be expressed as $S = \sum_{i,j=1}^{n} p_j x_{ij}$.

Let c_{ij} be the cost of the edge between vertices i and j. Using this notation, when inserting a vertex k between i and j, the total accumulated cost of the solution changes by $\Delta c_{ikj} = c_{ik} + c_{kj} - c_{ij}$.

We chose the heuristic value of an insertion move "vertex k between vertices i and j" to be $h_{ikj} = \frac{p_k}{\Delta c_{ikj}}$.

Generating the feasible insert moves is done as follows: for each vertex k that is not yet included in the solution, we calculate Δc_{ikj} for each pair of consecutive vertices in the solution (i, j). Any move for which the resulting cost of the solution

$c + \Delta c_{ikj} > d_{lim}$ where d_{lim} is the distance budget for the problem instance, is infeasible. In case of the GRASP-ILS hybrid, the vertices that have been eliminated in the perturbation step are also excluded from move generation.

Additionally, we implemented a path optimizing local search using `2-opt` moves. The search removes two edges in the solution and replaces them with two different edges, provided that the total path cost decreases. This is repeated until no moves that can decrease the total cost remain. This search is used by GRASP-PR as the local search function in the GRASP main body and after each path relinking step.

3.1.1 Comparison to published results

The comparison is made to the optimum or best published results. In case of the Tsiligirides (1984) instances we compare to the optimum results obtained with CPLEX (Souffriau 2010). In case of the 64- and 66-vertex instances of Chao et al. (1996b) we compare against the best published result (Silberholz and Golden 2010; Schilde et al. 2009). For the TSPLIB based instances the optimum value is taken from (Fischetti et al. 1998).

Table 3.1 gives the average minimum, arithmetic mean and maximum gap over each set of test instances. GRASP-PR is the cooperative GRASP with path relinking. GRILS-T is the cooperative GRASP-ILS hybrid using the trajectory rejoin technique. GRILS-I in the independent (non-cooperative) version of the same algorithm. We compare the results to the 2-PIA algorithm of Silberholz and Golden (2010) for which the results are available for all of the OP datasets included in the experiment. We've included the average time to solution t_{avg} for each algorithm. While direct comparisons between run times are not possible due to various factors, this gives an estimate of expected time to solution in a practical application.

2-PIA has the best solution quality on the smaller instances. For the larger TSPLIB based dataset of Fischetti et al., GRASP-T performed more consistently, having a better average and maximum gap. With the exception of the small Tsili-

Table 3.1: Summary of gap to optimum in OP benchmarks. 0% gap indicates optimum results, values larger than 0% indicate suboptimal results. Best results are in bold

dataset	*instances*		*Min.*	*Avg.*	*Max.*	t_{avg}, *s*
Tsiligirides	49	2-PIA	**0.00%**	**0.00%**	**0.00%**	0.21
		GRILS-T	**0.00%**	0.00%	0.05%	0.04
		GRILS-I	**0.00%**	**0.00%**	**0.00%**	0.05
		GRASP-PR	**0.00%**	0.13%	0.30%	1.5
Chao	40	2-PIA	0.38%	**0.38%**	**0.38%**	0.76
		GRILS-T	**0.06%**	0.48%	1.04%	0.40
		GRILS-I	0.24%	0.59%	1.03%	1.3
		GRASP-PR	1.62%	3.13%	4.14%	3.3
Fischetti	123	2-PIA	**1.66%**	3.62%	4.31%	7.1
		GRILS-T	1.70%	**2.86%**	**4.01%**	4.1
		GRILS-I	2.40%	3.29%	4.09%	21
		GRASP-PR	1.88%	3.25%	4.68%	44

girides instances, where GRASP-I has produced the optimum result in each test run, it also dominates the other two evaluated parallelization approaches both in terms of solution quality and average time to solution.

Appendix A.1 contains further detail about the experiment, including the gap results for GRILS-T in each problem instance.

3.1.2 Execution time

For a randomized algorithm, timing of a single test run does not give an accurate prediction of expected running time to a sufficiently high quality solution. We performed additional tests of 100 runs on selected problem instances. The tests were executed with the same number of workers (23) as the tests used to measure solution quality (Section 3.1.1).

The criteria for choosing the problem instances for this test were as follows:

- all algorithms used in the experiment solved the instance to optimality (or best known result) at least once;

- two or more test runs (of any algorithm) reached the solution slower than 1 seconds;
- the problems with lower number of vertices were chosen among the possible candidates.

These criteria were aimed to reduce the computation time required to produce the results, at the same time avoiding trivial instances that can be solved, for example, by greedy construction alone.

Using the methodology proposed by Aiex et al. (2002), we present the results as an empirical probability distribution of "time to target value". We sort tests $i = 1, ..., 100$ by the wall clock time t_i spent to reach the solution with a score equal or better than the target. The probability that the target is reached at a given time t_i is then $p_i = \frac{i-0.5}{100}$.

Figure 3.1 shows the empirical distribution of time to target value for the 48-vertex problem instance `gr48`. Figure 3.2 shows the distribution for another 48-vertex problem, `hk48`. 2% gap was selected as target in both cases. The graphs show that for these problems, the probability of reaching a high quality solution after 1 second of runtime exceeds 0.8 when using GRILS-T and GRILS-I.

3.1.3 Parallel performance

We evaluate the parallel performance by the efficiency metric. Ideal, or linear speedup is achieved when the execution time of the algorithm is reduced n times when executed by n workers in parallel. This corresponds to efficiency of 1. Let T_1 be the average time to reach a target value for a problem instance with one worker process. Efficiency $E_n = \frac{T_1}{nT_n}$.

To measure parallel efficiency, we performed tests by using from 1 to 128 worker processes on problem instances `gr48` and `hk48` that were selected for the execution time experiment in Section 3.1.2. These test runs were not node-local, i.e. node-to-node network communication was used. We measured wall clock time to reach a target solution of 5% gap or better over 20 test runs.

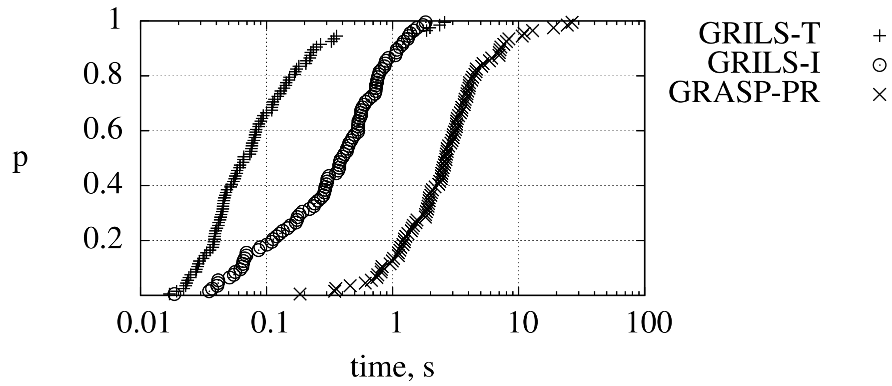

Figure 3.1: Probability distribution of time to 2% gap for OP instance `gr48` (Generation 1).

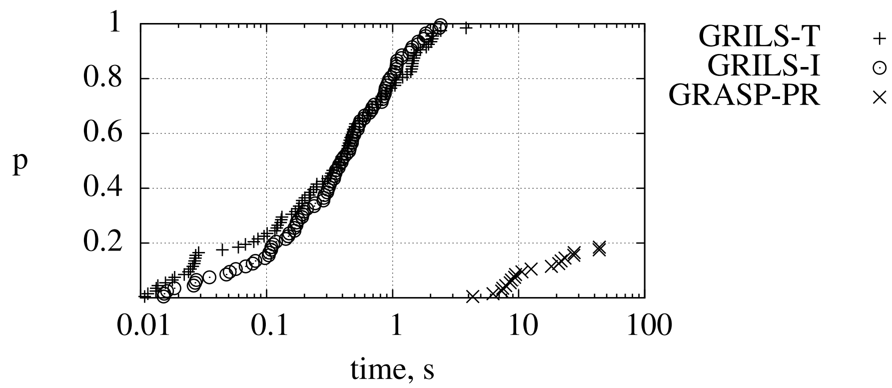

Figure 3.2: Probability distribution of time to 2% gap for OP instance `hk48` (Generation 1).

Table 3.2: Average efficiency of the parallelization approaches over the selected OP instances, by number of parallel workers. Efficiency of 1 is equivalent to linear speedup. Values between 0 and 1 indicate that some of the computing resource is used redundantly, spent in communication or in blocking waits.

	workers						
	2	*4*	*8*	*16*	*32*	*64*	*128*
GRILS-T	1.15	1.96	1.37	1.38	0.91	0.73	0.23
GRILS-I	1.00	0.93	0.83	1.02	0.68	0.51	0.35
GRASP-PR	-	-	-	-	-	-	-

Table 3.2 gives the average efficiency with 2 to 128 worker processes over the selected problem instances. The difference between cooperative (GRILS-T) and non-cooperative (GRILS-I) versions of the GRASP-ILS hybrid even out starting from 32 parallel processes. Overall both approaches display excellent scalability with to up to 64 processes. GRASP with path relinking was not capable of producing results with 5% or better gap consistently with one process. Because we did not obtain a reliable measure of T_1 for GRASP-PR, it was omitted from the comparison.

3.2 Generalized Orienteering Problem

While the GOP can be interpreted to be the OP with any objective function and any cost function (Silberholz and Golden 2010; Ramesh and Brown 1991), we implemented the formulation that is commonly referred to as the GOP in the literature. In this formulation, a vertex j has m attributes with associated scores $(p_{j1}, p_{j2}, \ldots, p_{jm})$. A weight vector $W = (w_1, w_2, \ldots, w_m)$ assigns importance to each attribute. The objective function $S = \sum_{a=1}^{m} w_a \left(\sum_{i,j=1}^{n} p_{ja}^{k} x_{ij} \right)^{\frac{1}{k}}$. As defined in Section 3.1, x_{ij} is 1 if the edge from i to j is included in the solution and 0 otherwise.

The heuristic function was implemented as follows. Let S be the value of the objective function for the current solution. Let S_k be the objective function calculated for the solution where a new vertex k is inserted. The heuristic value for an insertion move between vertices i and j is $h_{ikj} = \frac{S_k - S}{\Delta c_{ikj}}$. The move is evaluated based on the change in total score after the insertion. This is equal in predictive power to the heuristic used for the OP.

In other respects the problem is identical to the OP. The feasible move generation and local search were implemented as described in Section 3.1.

Table 3.3: Summary of gap to optimum in GOP benchmarks. 0% gap indicates optimum results, values larger than 0% indicate suboptimal results. Best results are in bold

	Min.	*Avg.*	*Max.*	t_{avg}, *s*
2-PIA	**0.00%**	**0.00%**	**0.00%**	0.46
GRILS-T	0.22%	0.64%	1.39%	0.46
GRILS-I	0.33%	0.56%	0.91%	0.76
GRASP-PR	0.01%	0.11%	0.20%	3.33

3.2.1 Comparison to published results

We evaluated the algorithms on a 27-vertex problem dataset representing cities in China. The reader is referred to the paper of Wang et al. (2008) for the full description of the "Chinese Cities" benchmark and Appendix A.2 for the parameters and detailed results of our experiment.

We compare the results of the experiments to the results for the two-parameter iterative algorithm (2-PIA) of Silberholz and Golden (2010). Table 3.3 lists the average gap to the best known solution and the average runtime to the best solution over all test runs. Despite the small number of vertices in the graph, none of the parallelization approaches were capable of reproducing the performance of 2-PIA. The GRASP with path relinking is the only algorithm that uses `2-opt` local search and appears to benefit from that. The run times of the GRASP-ILS hybrid approaches are comparable to that of 2-PIA. Overall this comparison favours 2-PIA, which ran on a single core, to our parallel algorithms that used 23 virtual cores.

3.2.2 Execution time

We calculate the empirical probability distribution of the random variable "time to target value" (see Section 3.1.2 for the description of the methodology). We chose two sets of parameters for the experiment, based on the results of the solution quality benchmark:

- all algorithms used in the experiment solved the instance to the best known result at least once;

- two or more test runs (of any algorithm) reached the solution slower than 1 seconds;
- $k > 1$ ($k = 1$ is equivalent to the OP).

We set target gap to 0.5%, because with parameter $k \geq 3$ the numerical differences between solution scores are small. Figure 3.3 shows the empirical probability for $k = 3$, with the weight vector $W = (0, 1, 0, 0)$. In Figure 3.4 $k = 4$. All of the evaluated algorithms display similar performance. As was the case with the OP, the target solution can be expected to be reached by 1 second of computation with the probability of 0.8 or better for the GRASP-ILS hybrid. GRASP with path relinking was marginally slower.

3.2.3 Parallel performance

Scalability on the GOP dataset was evaluated with the same parameters that were used in the execution time experiment. We measured wall clock time to target solution that is equal or better than 2% gap to the best known solution for the given parameter set. In Table 3.4 the average efficiency over 20 test runs for the two parameter sets is given for the tested number of processors.

The efficiency of both parallelization methods of the GRASP-ILS hybrid is excellent, although it can be observed to drop with 64 and 128 parallel workers. Meanwhile, our implementation of GRASP with path relinking is clearly inefficient and does not scale according to expectations.

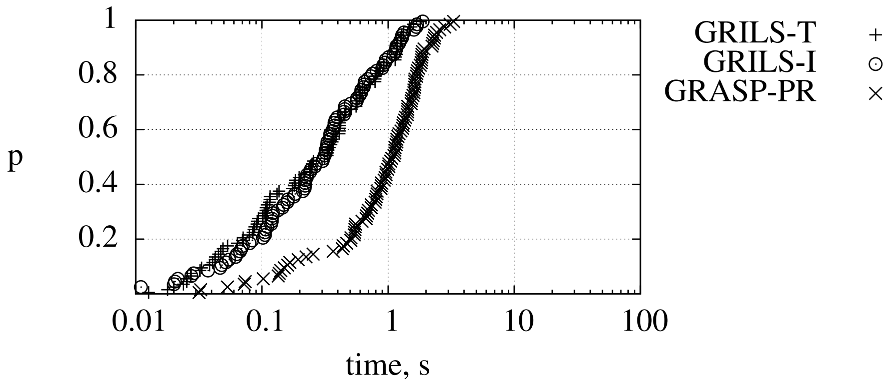

Figure 3.3: Probability distribution of time to 0.5% gap in the GOP benchmark ($k = 3$, weight vector $W = (0, 1, 0, 0)$).

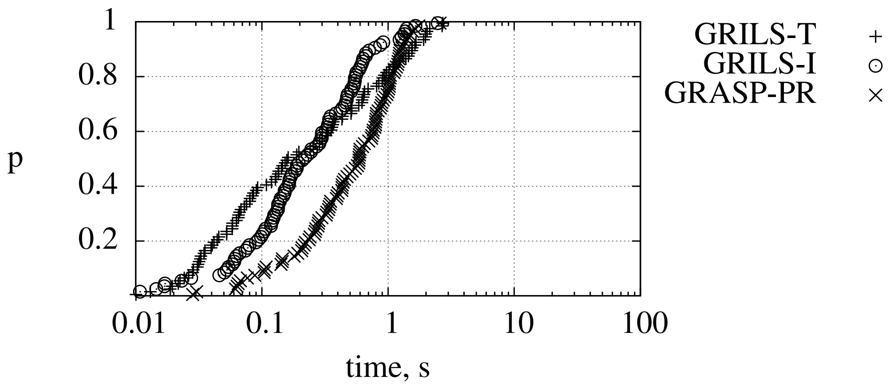

Figure 3.4: Probability distribution of time to 0.5% gap in the GOP benchmark ($k = 4$, weight vector $W = (0, 1, 0, 0)$).

Table 3.4: Average efficiency of the parallelization approaches over the selected GOP instances, by number of parallel workers. Efficiency of 1 is equivalent to linear speedup. Values between 0 and 1 indicate that some of the computing resource is used redundantly, spent in communication or in blocking waits.

	workers						
	2	*4*	*8*	*16*	*32*	*64*	*128*
GRILS-T	1.02	0.89	1.10	0.94	0.67	0.40	0.24
GRILS-I	0.90	0.78	0.58	0.60	0.57	0.30	0.21
GRASP-PR	1.08	0.57	0.27	0.08	0.02	0.01	0.00

3.3 Team Orienteering Problem with Time Windows

The team orienteering problem (TOP) is a generalization of the OP where the solution consists of a predetermined number of $m \geq 1$ tours. Each tour must honor the distance limit d_{lim}. A vertex cannot be included in more than one tour. When extended with time windows (TOPTW), a vertex k has an opening time O_k and a closing time C_k. Let P_k be the set of vertices preceding k in the solution and including k itself. The arrival time at vertex k is formally calculated as $a_k = \max\{O_k, \sum_{i=1}^{n} \sum_{j \in P_k} x_{ij} c_{ij}\}$. The solution is considered feasible if $a_k \leq C_k$ for all vertices k that are part of the solution. We allow waiting at the vertex if the opening time has not yet arrived.

Addressing multiple tours in feasible move generation is straightforward. When considering the insertion of a vertex k, all possible insertion points in all tours are generated as moves. Supporting time windows efficiently requires more work, because insertions and removals cause changes in the arrival times of the vertices succeeding the affected position in the solution. We adopt the approach of Vansteenwegen et al. (2009). The time shift s_k is calculated for a move of inserting vertex k into the solution. In the theoretical model the vertices have no cost associated, so for the insertion of k between vertices i and j, we may express the time shift $s_{ikj} = c_{ik} + w_k + c_{kj} - c_{ij}$ where $w_k = \max\{0, O_k - a_k\}$ is the waiting time at k.

Each vertex k in the solution is associated with the maximum legal shift ("maxshift") $s_{max_k} = \min\{C_k - a_k, w_j + s_{max_j}\}$ where j denotes the vertex following k. To evaluate the feasibility of an insertion, the algorithm needs to check that the condition $s_{ikj} \leq s_{max_j}$ is fulfilled.

The maxshift variables need to updated on each insertion and removal. However, the time complexity of this operation is $O(n^2)$ for the solution size n per each iteration. Since the feasible move generation already is of polynomial complexity, the update of maxshift does not significantly reduce the performance of the search.

As the objective function of TOPTW is the same as the objective function of OP, we can derive the heuristic function to evaluate each insertion with a minor modification. $h_{ikj} = \frac{p_k}{s_{ikj}}$, because the calculated shift directly expresses how much cost the move inflicts on the solution.

With the introduction of time windows, `2-opt` based local search becomes less effective as it causes reversals of sub-paths in the solution (Potvin and Robillard 1995). We have omitted our local search completely from the experiments involving problems with time windows.

3.3.1 Comparison to published results

The TOPTW experiments were carried out on the dataset of Montemanni and Gambardella (2009). The comparison was made to two previously published algorithms. Their average run times and gap values are taken from (Cura 2014). The iterated local search (ILS) of Vansteenwegen et al. (2009) is one of the earliest algorithms to tackle this problem. It has also remained one of the fastest algorithms according to the literature. Later publications have improved on the results of ILS, which invariably has involved a compromise in speed. Among those, the iterative three-component heuristic (I3CH) has reached very high solution quality, especially when the number of tours $m > 1$. Those results were obtained by using, on the average, 3 minutes and 40 seconds of computation (Hu and Lim 2014). A more recent paper of Qin et al. (2015) reports a marginally better average gap by using tabu search with the average computation time of 3 minutes and 42 seconds.

We present the results, organized by the dataset and the number of tours, in Table 3.5 and Table 3.6. For the previously published results, we've kept the numerical precision of Cura (2014), except for I3CH for Solomon instances with 4 tours, where the gap was very small but not equal to 0.

ILS is a deterministic algorithm so the *Min.*, *Avg.* and *Max.* columns contain the same values. I3CH contains nondeterministic components but the published results were obtained using only one test on each instance, which is why the minimum, maximum and mean values are also taken to be equal.

The results of the cooperative GRASP-ILS hybrid compare favourable to those of ILS. The average gap is better in all cases. The execution times are not directly

Table 3.5: Summary of gap to optimum in TOPTW benchmarks (Solomon instances). m is the number of tours in the benchmark. 0% gap indicates optimum results, values larger than 0% indicate suboptimal results. Best results are in bold

tours	*instances*		*Min.*	*Avg.*	*Max.*	t_{avg}, *s*
$m = 1$	55	ILS	2.3%	2.3%	2.3%	0.9
		I3CH	0.9%	0.9%	**0.9%**	79
		GRILS-T	**0.24%**	**0.66%**	1.13%	1.9
		GRILS-I	0.51%	0.81%	1.13%	5.0
		GRASP-PR	1.07%	1.70%	2.33%	8.5
$m = 2$	55	ILS	2.4%	2.4%	2.4%	1.7
		I3CH	**0.3%**	**0.3%**	**0.3%**	266
		GRILS-T	0.86%	1.46%	2.08%	2.9
		GRILS-I	1.15%	1.59%	1.98%	9.2
		GRASP-PR	2.16%	2.98%	3.65%	16
$m = 3$	55	ILS	1.8%	1.8%	1.8%	1.6
		I3CH	**0.1%**	**0.1%**	**0.1%**	89
		GRILS-T	0.90%	1.42%	1.96%	2.9
		GRILS-I	1.02%	1.49%	1.88%	7.1
		GRASP-PR	1.92%	2.66%	3.32%	14
$m = 4$	55	ILS	1.7%	1.7%	1.7%	1.7
		I3CH	**0.03%**	**0.03%**	**0.03%**	107
		GRILS-T	1.08%	1.64%	2.21%	1.7
		GRILS-I	1.35%	1.76%	2.06%	5.0
		GRASP-PR	2.00%	2.61%	3.11%	7.7

Table 3.6: Summary of gap to optimum in TOPTW benchmarks (Cordeau instances). m is the number of tours in the benchmark. 0% gap indicates optimum results, values larger than 0% indicate suboptimal results. Best results are in bold

tours	*instances*		*Min.*	*Avg.*	*Max.*	t_{avg}, *s*
		ILS	7.4%	7.4%	7.4%	1.9
		I3CH	2.4%	**2.4%**	**2.4%**	120
$m = 1$	20	GRILS-T	**1.30%**	2.76%	4.09%	2.6
		GRILS-I	2.18%	3.25%	4.26%	9.0
		GRASP-PR	2.70%	5.40%	7.61%	11
		ILS	7.0%	7.0%	7.0%	5.0
		I3CH	**1.3%**	**1.3%**	**1.3%**	276
$m = 2$	20	GRILS-T	2.45%	4.66%	6.83%	4.5
		GRILS-I	4.42%	5.72%	6.76%	20
		GRASP-PR	5.47%	7.74%	9.63%	32
		ILS	8.3%	8.3%	8.3%	9.5
		I3CH	**0.4%**	**0.4%**	**0.4%**	461
$m = 3$	20	GRILS-T	3.49%	5.65%	7.52%	5.8
		GRILS-I	5.55%	6.66%	7.60%	26
		GRASP-PR	6.22%	7.95%	9.45%	56
		ILS	8.2%	8.2%	8.2%	13.9
		I3CH	**0.1%**	**0.1%**	**0.1%**	648
$m = 4$	20	GRILS-T	4.20%	5.95%	7.36%	6.9
		GRILS-I	5.93%	6.88%	7.65%	35
		GRASP-PR	6.45%	7.63%	8.78%	79

Table 3.7: New best TOPTW results found by the cooperative GRASP-ILS hybrid

instance	m	*new best*
rc208	1	1046
r104	2	550
r107	2	538
pr13	1	467
pr09	2	897

comparable because of the different hardware configurations, but they indicate that in applications where the tour needs to be computed in response to an online query, both algorithms in their test configuration would be equally usable.

The computation time of I3CH is much larger, which would make it more suitable for pre-computing or batch processing tours. Hu and Lim (2014) have reported that they also obtain high quality solutions with lower computation time as well. However, these computation times are still on the average more than 2 times slower (7.5 seconds in the fastest benchmark) than those of ILS and GRILS-T, while the solutions quality at $m = 1$ is worse than our GRILS-T and approaching our results at $m = 2$. I3CH remains superior at $m > 2$.

The other two parallelization approaches that were used in the experiment were less successful. In terms of average solution quality, GRILS-I is dominated by GRILS-T, although the difference is under 0.2% on Solomon instances and approximately 1% on Cordeau instances. The difference is mainly in the time it takes to reach the solutions, with the independent version being 5 times slower. Our implementation of GRASP with path relinking was inferior to the rest of the algorithms in both solution quality and speed.

The cooperative GRASP-ILS hybrid also found new best solutions in 5 of the instances (Table 3.7). The full results and details of the experiment are given in Appendix A.3.

3.3.2 Execution time

We use empirical probability distribution of ”time to target value” to estimate how long it takes for each of the presented parallelization approaches to reach a high quality solution. We performed 20 tests on two TOPTW instances, chosen using the following criteria:

- all algorithms used in the experiment solved the instance to the best known result at least once;
- two or more test runs (of any algorithm) reached the solution slower than 1 seconds;
- the problems with lower number of tours were preferred.

These tests were performed with 23 parallel workers and node-locally. The results vary by the chosen instance. `rc108` was solved significantly faster, with GRASP-T producing the required solution with probability 0.9 in 0.1 seconds (Figure 3.5). For `r101` (two tours), it took approximately a second for both GRASP-ILS hybrids to produce a high quality solution with the probability 0.8 or higher (Figure 3.6).

3.3.3 Parallel performance

The scalability of the GRASP-ILS hybrids is very good with up to 32 parallel workers, where the efficiency is near 1 (Table 3.8). With 64 and 128 workers some degradation in efficiency appears. The tests were performed on `rc108` ($m = 1$) and `r101` ($m = 2$), with 5 test runs for each combination of the instance and the number of workers. With GRASP-PR we were unable to obtain a reliable measure of the average time with 1 worker at target gap 5% and therefore unable to calculate the speedup.

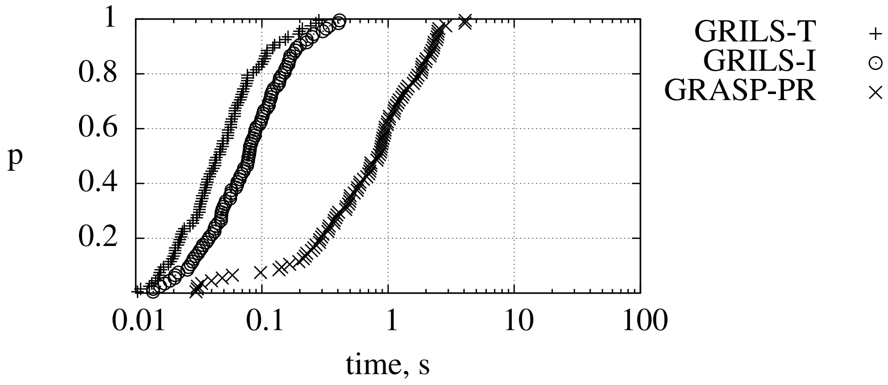

Figure 3.5: Probability distribution of time to 2% gap for the TOPTW instance `rc108` ($m = 1$).

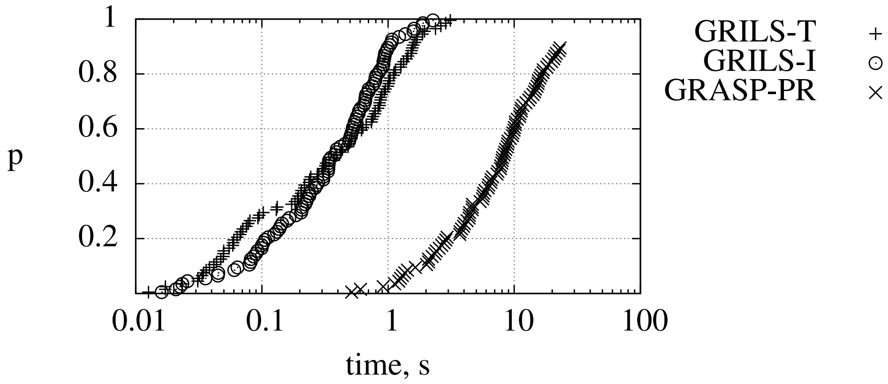

Figure 3.6: Probability distribution of time to 2% gap for the TOPTW instance `r101` ($m = 2$).

Table 3.8: Average efficiency of the parallelization approaches over the selected TOPTW instances `rc108` ($m = 1$) and `r101` ($m = 2$), by number of parallel workers. Efficiency of 1 is equivalent to linear speedup. Values between 0 and 1 indicate that some of the computing resource is used redundantly, spent in communication or in blocking waits.

	workers						
	2	*4*	*8*	*16*	*32*	*64*	*128*
GRILS-T	1.10	0.90	1.08	1.01	0.92	0.51	0.25
GRILS-I	1.39	1.04	1.24	1.21	0.81	0.63	0.33
GRASP-PR	-	-	-	-	-	-	-

3.4 Multi-Constrained Team Orienteering Problem with Time Windows

In this experiment, we used the variation of the problem with multiple time windows (MCTOPMTW). The differences from TOPTW (Section 3.3) are as follows:

- each vertex k has multiple pairs of opening and closing times $\{(O_{k_1}, C_{k_1}), \ldots, (O_{k_n}, C_{k_n})\}$.
- each vertex has a type vector $\langle T_1, T_2, \ldots, T_n \rangle$ where T_i is 1 if the vertex is of the given type and 0 otherwise. There is also a global constraint T_{max_i} of the number of each type i in the solution P, so that $\sum_{j \in P} T_j(i) \leq T_{max_i}$.
- each vertex has one or more associated attributes $\{A_1, \ldots A_n\}$ which have numerical values and must honor a global constraint $A_i \leq A_{max_i}$.

We implement the type count and attribute constraints by keeping track of the number of each type and sum of the attributes over all of the vertices in the solution. An insertion move is discarded, if the vertex would cause one of the constraints to be violated. If this happens, the vertex is ignored in move generation until some vertices are removed from the solution (the perturbation phase or resetting the solution).

When performing an insertion or removal move or generating insertion moves, the current time window is determined first. This is the earliest time window for which $C_{k_i} < a_k$ (k refers to the vertex, a_k is the arrival time at the vertex and i is the index of the opening/closing time pair). After this, the current time window is used in calculations in the same way as the single time window with the TOPTW.

Because the objective function and the computation of shifts is the same as with the TOPTW, we may use the same heuristic function as with the TOPTW (Section 3.3).

3.4.1 Comparison to published results

We compare our parallelization approaches to the sequential solution of Souffriau et al. (2013). Their GRASP-ILS hybrid differs from ours in implementation details but has a similar structure. To the best of our knowledge, their average gap values are also the best published results. The dataset used in the experiment is derived from the TOPTW datasets that are widely used. See Appendix A.4 for details.

We have listed the results of Souffriau et al. (2013) with one modification. They give the total runtime over 10 test runs. In our comparisons we have used average runtime per one test run, so we divide their run times by 10.

Our results are clearly inferior to the state of the art, although in case of the Cordeau et al. based instances and one tour, GRILS-T has reached better average solution quality (Table 3.9).

3.4.2 Execution time

We estimate the executing time by determining the probability distribution of the random variable ”time to target value” empirically. Two problem instances where selected using the same guiding principles as in the TOPTW experiment (Section 3.3.2). 20 test runs were performed with 23 parallel workers.

Figure 3.7 shows the probability curve for the instance `r106`. In Figure 3.8 the instance `rc108` is shown. In both cases, the number of tours $m = 1$ and the target gap chosen was 2%. Both GRASP-ILS hybrid approaches reached the selected gap in under 0.5 seconds, with probability 0.8 or higher.

3.4.3 Parallel performance

The scalability tests were done on the same instances as the probability distribution experiment. We performed 5 test runs with the number of parallel workers varying from 1 to 128. The efficiency of the GRASP-ILS hybrids is excellent in all the tested configurations. In these tests both parallelizations achieved near-linear speedup (Table 3.10). We omit the results for GRASP with path relinking, as we were unable to reliably measure the solution time to the target 5% gap with one worker.

Table 3.9: Summary of gap to optimum in MCTOPMTW benchmarks. m is the number of tours in the benchmark. 0% gap indicates optimum results, values larger than 0% indicate suboptimal results. Best results are in bold

dataset	*instances*		*Min.*	*Avg.*	*Max.*	t_{avg}, *s*
Solomon ($m = 1$)	29	GRASP-ILS	**0.59%**	**1.21%**	2.02%	0.27
		GRILS-T	1.75%	1.77%	1.92%	0.27
		GRILS-I	1.75%	1.77%	**1.89%**	0.33
		GRASP-PR	2.30%	2.89%	4.07%	1.8
Solomon ($m = 2$)	29	GRASP-ILS	**1.30%**	**2.55%**	**3.85%**	0.77
		GRILS-T	1.92%	3.47%	5.38%	1.7
		GRILS-I	2.31%	3.76%	4.99%	4.3
		GRASP-PR	2.91%	5.40%	7.36%	6.8
Solomon ($m = 3$)	29	GRASP-ILS	**1.80%**	**3.56%**	**4.96%**	1.4
		GRILS-T	3.47%	5.95%	8.17%	2.9
		GRILS-I	5.17%	6.59%	7.96%	8.2
		GRASP-PR	5.67%	7.96%	9.64%	11
Solomon ($m = 4$)	29	GRASP-ILS	**2.85%**	**4.28%**	**5.59%**	2.3
		GRILS-T	4.60%	7.04%	8.88%	3.3
		GRILS-I	6.17%	7.71%	8.84%	11
		GRASP-PR	7.23%	9.24%	10.67%	14
Cordeau ($m = 1$)	8	GRASP-ILS	4.24%	5.86%	8.14%	0.68
		GRILS-T	**1.41%**	**1.94%**	**5.01%**	2.3
		GRILS-I	2.28%	3.80%	5.04%	7.4
		GRASP-PR	4.45%	7.67%	10.87%	7.9
Cordeau ($m = 2$)	8	GRASP-ILS	**1.91%**	**3.94%**	**5.55%**	1.8
		GRILS-T	4.96%	7.70%	11.06%	2.9
		GRILS-I	7.61%	9.23%	10.52%	15
		GRASP-PR	7.03%	10.22%	12.94%	18
Cordeau ($m = 3$)	8	GRASP-ILS	**2.57%**	**4.38%**	**5.85%**	3.2
		GRILS-T	5.25%	7.64%	10.03%	5.8
		GRILS-I	7.43%	8.89%	10.13%	24
		GRASP-PR	8.38%	10.09%	11.65%	33
Cordeau ($m = 4$)	8	GRASP-ILS	**2.86%**	**4.18%**	**5.24%**	5.2
		GRILS-T	5.43%	7.21%	8.67%	7.6
		GRILS-I	6.86%	8.02%	9.10%	33
		GRASP-PR	6.86%	8.60%	9.88%	50

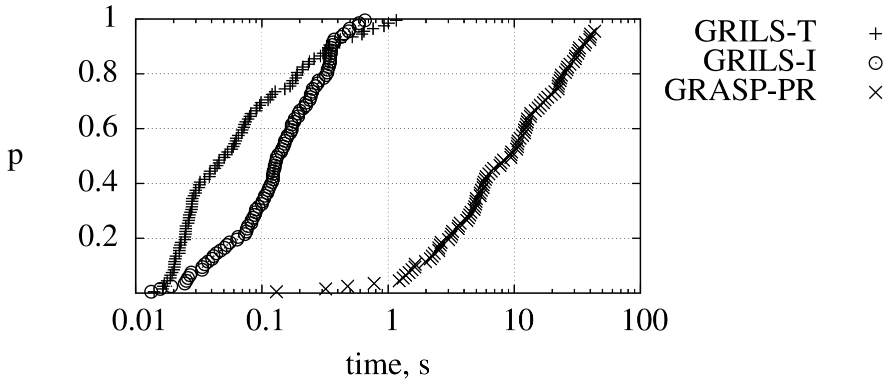

Figure 3.7: Probability distribution of time to 2% gap for the MCTOPMTW instance `r106` ($m = 1$).

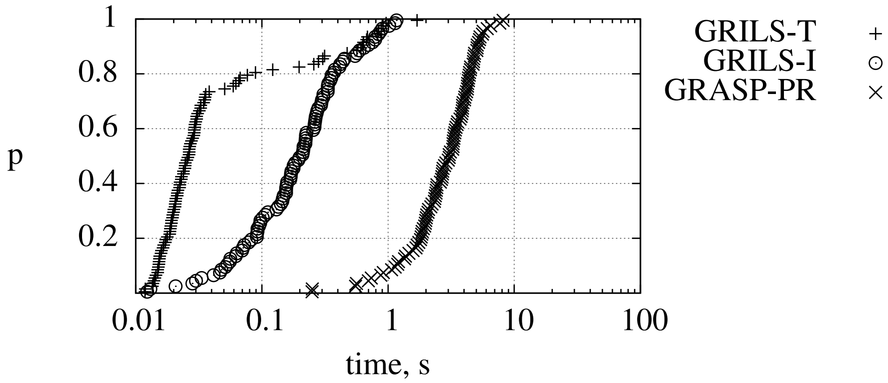

Figure 3.8: Probability distribution of time to 2% gap for the MCTOPMTW instance `rc108` ($m = 1$).

Table 3.10: Average efficiency of the parallelization approaches over the selected MCTOPMTW instances `r106` ($m = 1$) and `rc108` ($m = 1$), by number of parallel workers. Efficiency of 1 is equivalent to linear speedup. Values between 0 and 1 indicate that some of the computing resource is used redundantly, spent in communication or in blocking waits.

	workers						
	2	*4*	*8*	*16*	*32*	*64*	*128*
GRILS-T	0.86	0.73	0.97	1.28	1.82	1.16	1.35
GRILS-I	1.30	1.10	1.17	1.39	0.91	0.93	0.73
GRASP-PR	-	-	-	-	-	-	-

3.5 Time Dependent Orienteering Problem

The time dependent orienteering problem (TDOP) is a generalization of the OP where the cost of traveling an edge from vertex i to j is a function of the departure (because in the formalized OP, vertices have no cost, this is also the arrival) time from vertex i. Based on (Verbeeck et al. 2014), $c(i, j, a_i)$ is a piecewise linear function. The distances between vertices are fixed. The speed of travel is dependent on both the edge type and time of day. This simulates typical commute and urban traffic patterns.

An insertion or removal of a vertex is likely to cause a change in costs for the edges following it, because arrival times for the following vertices are shifted. Since the costs can both increase and decrease, there is a possibility that the total duration of the tour decreases on an insertion or increases on a removal.

The adaptation of our algorithms to TDOP includes efficient feasible move generation, the bookkeeping functionality to support that and a custom heuristic function. To efficiently check the feasibility of an insertion we adopt the "maxshift" approach of Verbeeck et al. (2014). For each vertex j, a'_j is the latest arrival time that does not cause the total duration of the tour to exceed the duration limit of the problem. The maximum allowed shift $s_{max_j} = a'_j - a_j$. For an insertion of vertex k before vertex j, we compute $s_{ikj} = c(i, k, a_i) + c(k, j, a_k) - c(i, j, a_i)$ and check that $s_{ikj} \leq s_{max_j}$ is satisfied.

In the perturbation phase of the GRASP-ILS hybrids we additionally need to check that the removal of a vertex does not destroy the feasibility of the solution. We use a simple heuristic measure of checking whether the vertex shortens the overall duration of the tour. For the removal of k to be allowed $s_{ikj} \geq 0$ must be satisfied.

The heuristic function $h_{ikj} = \frac{p_k}{\max\{\epsilon, s_{ikj}\}}$ where $\epsilon > 0$ is a small value that is used instead of a negative shift. For a constant p_k, h_{ijk} is then a monotonic function of s_{ijk} [1].

We do not use `2-opt` local search with TDOP as arbitrarily reversing subpath causes similar difficulties with the cost function as with the time windows

[1] although a strictly ordered mapping would be more accurate

Table 3.11: Summary of gap to optimum in TDOP benchmarks. 0% gap indicates optimum results, values larger than 0% indicate suboptimal results. Best results are in bold

	Min.	*Avg.*	*Max.*	t_{avg}, *s*
ACS	**0.2%**	**0.7%**	**1.3%**	0.1
GRILS-T	5.79%	8.78%	10.78%	5.8
GRILS-I	6.25%	8.00%	9.39%	7.3
GRASP-PR	5.97%	8.60%	10.88%	23

(Section 3.3).

3.5.1 Comparison to published results

Verbeeck et al. (2014) have divided their results in two parts. We tested our parallelization approaches with the smaller instances which have been solved to optimality with CPLEX[2]. This set includes 24 instances. The larger set which requires modification of edge costs based on intermediate CPLEX results was omitted from our comparison.

Table 3.11 shows a large disparity between both the speed and the solution quality of the ant colony system (ACS) of Verbeeck et al. and our approaches. This could indicate that our heuristic function is too inaccurate and the construction phase produces systemically flawed solutions, because in our experiment, both GRASP-ILS hybrids examined over 250000 solutions for each instance. This is much more than the ACS needed to arrive at its high quality results.

We've omitted the execution time and scalability results for TDOP. They would be less meaningful as both depend on the measurement of time to high quality target solution. The evaluated algorithms were unable to produce high quality solutions on this benchmark.

[2] `http://www.ibm.com/software/integration/optimization/cplex-optimizer/`

Chapter 4

Discussion

This research makes the following contributions:

- We have presented experimental results of a metaheuristic approach to solving generalizations of the OP in parallel.
- We have designed a parallelization technique for the GRASP-ILS hybrid metaheuristic.
- We have shown that the approach is scalable and adaptable to applications where the response time needs to be low.

The experiments cover four generalized models that are specifically chosen for their applicability in tourist trip design. According to Gavalas et al. (2014), parallelized approach to TTDP-s is a relevant yet unexplored research direction. The design of the algorithms has intentionally been generic, with problem-specific components plugged in as necessary. We covered these adaptations in Chapter 3 for each of the experiments.

Earlier parallelization techniques of the GRASP have exclusively relied on path relinking (Resende and Ribeiro 2010). The GRASP-ILS hybrid metaheuristic, that has already been shown to be effective in TTDP related problems (Souffriau et al. 2013) does not use an pool of elite solutions. We have adopted a simple technique of sharing best solutions between the parallel workers to concentrate the search efforts in the neighbourhood of good solutions. This can be described as rejoining search trajectories.

Distributed computing platforms that have expensive communication are more prevalent than highly parallel closely integrated systems. Such platforms include clusters and grids. In the environment of expensive messaging, non-cooperative approaches are a strong alternative. Throughout the experiment we also evaluated the performance of the independent parallel GRASP-ILS hybrid.

4.1 Application opportunities

Similar metaheuristics have already been deployed in recommendation systems like CityTripPlanner[1] and VisitEstonia[2]. The focus on the insertion move in the search and random sampling makes them successful in models with time windows and other constraints.

The greedy random construction can be conveniently started with a partial solution. In the TTDP case, the user may provide their partial itinerary, which GRASP can then augment. The POIs chosen by the user can be flagged so that they are not removed from the solution by the recommender. The same principle can also be applied in incremental, interactive construction of the solution.

Team variants of the OP have been used to model multi-day trips. This is useful for an elegant mathematical formulation. However, a single tour spanning multiple days is an equally valid approach, as it covers multiple OP related operations research problems. For example, accommodation can either be pre-selected as part of the input or be included among the vertices of the problem with suitable time windows associated. We suggest a single-tour model with multiple time windows and compulsory vertices as a practical approach to TTDP. The metaheuristic template we have presented is directly applicable to such a model.

[1] http://citytripplanner.be/

[2] http://www.visitestonia.com/en/travel-planner

4.2 Further study opportunities

Several options in improving the algorithms were left unexplored. In the perturbation phase of the GRASP-ILS hybrid, the β selection and the weighting of items to be eliminated can both be done using an approach similar to Reactive GRASP. β can also be chosen as a function of the current iteration number so that the perturbations decrease over time (as in the simulated annealing heuristic).

Sequential components of the algorithms can be improved by local search techniques that are suitable for the TOPTW model: `2-opt*` and `Or-opt` (Potvin and Robillard 1995; Mester and Bräysy 2005). Both of these are time window friendly, 2-opt* is specifically tailored for multiple tours.

Our implementation of GRASP-PR suffered in both solution quality and speed. This can be connected to its poor scalability, since the bulk of the experiments were carried out with 23 worker processes. Not enough effort went to improving its performance, so GRASP-PR cannot be discounted on the basis of our results. For example, assuming that the elite pool was over-contended, we can increase the locality by choosing the two-tiered elite pool strategy instead of a centralized approach and reducing the frequency of the messaging.

4.3 Conclusions

We have presented the parallelized GRASP-ILS hybrid. Out of the tested approaches, the cooperative parallel version (GRASP-T) reached the best gap values when tested on published benchmarks. It compared favourably to the state of the art on the team orienteering problem with time windows (TOPTW) benchmark, for which many competing algorithms exist.

Overall the solution quality results were mixed, most notably the performance of the algorithms was unsatisfactory on the time dependent orienteering problem (TDOP).

The independent version of the GRASP-ILS hybrid was slightly inferior overall. This indicates that our cooperative strategy is beneficial. Both parallelization methods of this metaheuristic displayed excellent scalability. In the tested configurations, they were capable of near-linear speedup. Both methods also have a characteristic probability distribution of time to target value, where high quality solutions have a high probability of appearing early in the search. This makes the techniques well suited in settings where the response time is a factor.

4.4 Acknowledgements

This research has been supported by EU through European Regional Development Fund.

Tallinn University of Technology provided the high performance computing (HPC) cluster platform used to running the experiments and mr. Toomas Kaevand gave useful assistance.

The author would like to thank prof. Tanel Tammet for providing the inspiration and motivation for this work.

Chapter 5

Summary

We presented a parallelization method for algorithms that solve a family of combinatorial optimization problems called the orienteering problem (OP) and its generalizations. This problem arises in the fields of logistics and automated planning and consists of finding a most profitable route between fixed locations, under time limit. An example of such a problem is automated creation of tourist trip plans, where the goal is to create a tour between places that are most interesting for the tourist, given that the tourist has a certain amount of time available.

We used an existing non-parallel hybrid metaheuristic that combines iterated local search (ILS) and the construction method of the greedy random adaptive search procedure (GRASP). We developed a novel parallelization method that benefits from cooperative behaviour. For comparison, we also implemented a non-cooperative parallel version of the same algorithm. To compare against a known parallel metaheuristic, we additionally implemented a version of the greedy randomized adaptive search procedure with path relinking (GRASP-PR) that follows the general guidelines published in the literature.

We compared our parallelization technique to these alternative versions to show that the cooperative approach contributes to the solution quality and speed and that our novel approach is competitive with a previously published approach. We performed tests with these three algorithms on five benchmark datasets from the literature. The results are also compared to non-parallel state of the art algorithms that have produced significant results on these benchmarks.

We showed that the our approach exhibits the useful property of reaching good quality suboptimal solutions early in the search, with predictable probability. Both cooperative and non-cooperative versions of the GRASP-ILS hybrid scaled well with up to 128 parallel worker processes. The quality of the solutions varied depending on the benchmark, with some problem types remaining difficult to solve.

Bibliography

Adomavicius, G., Manouselis, N., and Kwon, Y. Multi-criteria recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 769–803. Springer, 2011. ISBN 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3_24. URL `http://dx.doi.org/10.1007/978-0-387-85820-3_24`.

Aiex, R. and Resende, M. Parallel strategies for GRASP with path-relinking. In Ibaraki, T., Nonobe, K., and Yagiura, M., editors, *Metaheuristics: Progress as Real Problem Solvers*, volume 32 of *Operations Research/Computer Science Interfaces Series*, pages 303–333. Springer US, 2005. ISBN 978-0-387-25382-4. doi: 10.1007/0-387-25383-1_14. URL `http://dx.doi.org/10.1007/0-387-25383-1_14`.

Aiex, R. M., Resende, M. G. C., and Ribeiro, C. C. Probability distribution of solution time in GRASP: an experimental investigation. *J. Heuristics*, 8(3): 343–373, 2002. doi: 10.1023/A:1015061802659. URL `http://dx.doi.org/10.1023/A:1015061802659`.

Alba, E., Luque, G., and Nesmachnow, S. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48, 2013. ISSN 1475-3995. doi: 10.1111/j.1475-3995.2012.00862.x. URL `http://dx.doi.org/10.1111/j.1475-3995.2012.00862.x`.

Catalá, A., Martínez, J. J., and Mocholí, J. A. Strategies for accelerating ant colony optimization algorithms on graphical processing units. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007,* *25-28 September 2007, Singapore*, pages 492–500. IEEE, 2007. doi: 10.1109/CEC.2007.4424511. URL `http://dx.doi.org/10.1109/CEC.2007.4424511`.

Chao, I.-M., Golden, B. L., and Wasil, E. A. The team orienteering problem. *European Journal of Operational Research*, 88(3):464 – 474, 1996a. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(94)00289-4. URL `http://www.sciencedirect.com/science/article/pii/0377221794002894`.

Chao, I.-M., Golden, B. L., and Wasil, E. A. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475 – 489, 1996b. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(95)00035-6. URL `http://www.sciencedirect.com/science/article/pii/0377221795000356`.

Cordeau, J., Gendreau, M., and Laporte, G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997. doi: 10.1002/(SICI)1097-0037(199709)30:2⟨105::AID-NET5⟩3.0.CO;2-G. URL `http://dx.doi.org/10.1002/(SICI)1097-0037(199709)30:2<105::AID-NET5>3.0.CO;2-G`.

Cura, T. An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Computers & Industrial Engineering*, 74:270–290, 2014. doi: 10.1016/j.cie.2014.06.004. URL `http://dx.doi.org/10.1016/j.cie.2014.06.004`.

Erdogan, G. and Laporte, G. The orienteering problem with variable profits. *Networks*, 61(2):104–116, 2013. doi: 10.1002/net.21496. URL `http://dx.doi.org/10.1002/net.21496`.

Feo, T. A., Resende, M. G. C., and Smith, S. H. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994. doi: 10.1287/opre.42.5.860. URL `http://dx.doi.org/10.1287/opre.42.5.860`.

Fischetti, M., González, J. J. S., and Toth, P. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10(2):133–148, 1998. doi: 10.1287/ijoc.10.2.133. URL `http://dx.doi.org/10.1287/ijoc.10.2.133`.

Fomin, F. V. and Lingas, A. Approximation algorithms for time-dependent orienteering. *Inf. Process. Lett.*, 83(2):57–62, 2002. doi: 10.1016/S0020-0190(01)00313-1. URL `http://dx.doi.org/10.1016/S0020-0190(01)00313-1`.

García, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., and Linaza, M. T. Integrating public transportation in personalised electronic tourist guides. *Computers & OR*, 40(3):758–774, 2013. doi: 10.1016/j.cor.2011.03.020. URL `http://dx.doi.org/10.1016/j.cor.2011.03.020`.

Gavalas, D., Konstantopoulos, C., Mastakas, K., and Pantziou, G. E. A survey on algorithmic approaches for solving tourist trip design problems. *J. Heuristics*, 20(3):291–328, 2014. doi: 10.1007/s10732-014-9242-5. URL `http://dx.doi.org/10.1007/s10732-014-9242-5`.

Geem, Z. W., Tseng, C., and Park, Y. Harmony search for generalized orienteering problem: Best touring in china. In Wang, L., Chen, K., and Ong, Y., editors, *Advances in Natural Computation, First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings, Part III*, volume 3612 of *Lecture Notes in Computer Science*, pages 741–750. Springer, 2005. ISBN 3-540-28320-X. doi: 10.1007/11539902_91. URL `http://dx.doi.org/10.1007/11539902_91`.

Gionis, A., Lappas, T., Pelechrinis, K., and Terzi, E. Customized tour recommendations in urban areas. In Carterette, B., Diaz, F., Castillo, C., and Metzler, D., editors, *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 313–322. ACM, 2014. ISBN 978-1-4503-2351-2. doi: 10.1145/2556195.2559893. URL `http://doi.acm.org/10.1145/2556195.2559893`.

Hasuike, T., Katagiri, H., Tsubaki, H., and Tsuda, H. Tour planning for sightseeing with time-dependent satisfactions of activities and traveling times. *American Journal of Operations Research*, 3(3):369–379, 2013. doi: 10.4236/ajor.2013.33034. URL http://www.scirp.org/journal/PaperInformation.aspx?PaperID=31677.

Hasuike, T., Katagiri, H., Tsubaki, H., and Tsuda, H. Sightseeing route planning problem by electric vehicle on the time-expanded network. In *Computational Intelligence and Applications (IWCIA), 2014 IEEE 7th International Workshop on*, pages 147–152, Nov 2014. doi: 10.1109/IWCIA.2014.6988095.

Hu, Q. and Lim, A. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 232(2):276–286, 2014. doi: 10.1016/j.ejor.2013.06.011. URL http://dx.doi.org/10.1016/j.ejor.2013.06.011.

Labadi, N., Melechovský, J., and Calvo, R. W. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *J. Heuristics*, 17(6):729–753, 2011. doi: 10.1007/s10732-010-9153-z. URL http://dx.doi.org/10.1007/s10732-010-9153-z.

Marler, R. and Arora, J. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004. ISSN 1615-147X. doi: 10.1007/s00158-003-0368-6. URL http://dx.doi.org/10.1007/s00158-003-0368-6.

Mester, D. I. and Bräysy, O. Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers & OR*, 32:1593–1614, 2005. doi: 10.1016/j.cor.2003.11.017. URL http://dx.doi.org/10.1016/j.cor.2003.11.017.

Mocholí, J. A., Martínez, J. J., and Canós, J. H. A grid ant colony algorithm for the orienteering problem. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*, pages 942–949.

IEEE, 2005. ISBN 0-7803-9363-5. doi: 10.1109/CEC.2005.1554784. URL http://dx.doi.org/10.1109/CEC.2005.1554784.

Montemanni, R. and Gambardella, L. An ant colony system for team orienteering problems with time windows. *Foundations of Computing and Decision Sciences*, Vol. 34, No. 4:287–306, 2009.

Pietz, J. and Royset, J. O. Generalized orienteering problem with resource dependent rewards. *Naval Research Logistics (NRL)*, 60(4):294–312, 2013. ISSN 1520-6750. doi: 10.1002/nav.21534. URL http://dx.doi.org/10.1002/nav.21534.

Potvin, J. and Robillard, C. Clustering for vehicle routing with a competitive neural network. *Neurocomputing*, 8(2):125–139, 1995. doi: 10.1016/0925-2312(94)00012-H. URL http://dx.doi.org/10.1016/0925-2312(94)00012-H.

Prais, M. and Ribeiro, C. C. Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176, 2000. doi: 10.1287/ijoc.12.3.164.12639. URL http://dx.doi.org/10.1287/ijoc.12.3.164.12639.

Qin, H., Ming, W., Zhang, Z., Xie, Y., and Lim, A. A tabu search algorithm for the multi-period inspector scheduling problem. *Computers & OR*, 59(0): 78 – 93, 2015. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2015.01.003. URL http://www.sciencedirect.com/science/article/pii/S0305054815000106.

Ramesh, R. and Brown, K. M. An efficient four-phase heuristic for the generalized orienteering problem. *Computers & OR*, 18(2):151–165, 1991. doi: 10.1016/0305-0548(91)90086-7. URL http://dx.doi.org/10.1016/0305-0548(91)90086-7.

Resende, M. and Ribeiro, C. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In Gendreau, M. and Potvin, J.-

Y., editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 283–319. Springer US, 2010. ISBN 978-1-4419-1663-1. doi: 10.1007/978-1-4419-1665-5_10. URL http://dx.doi.org/10.1007/978-1-4419-1665-5_10.

Resende, M. G. and Ribeiro, C. C. *Parallel Greedy Randomized Adaptive Search Procedures*, pages 315–346. John Wiley & Sons, Inc., 2005. ISBN 9780471739388. doi: 10.1002/0471739383.ch14. URL http://dx.doi.org/10.1002/0471739383.ch14.

Ribeiro, C. C. and Rosseti, I. Efficient parallel cooperative implementations of GRASP heuristics. *Parallel Computing*, 33(1):21–35, 2007. doi: 10.1016/j.parco.2006.11.007. URL http://dx.doi.org/10.1016/j.parco.2006.11.007.

Ribeiro, C. C. and Urrutia, S. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, 179(3):775–787, 2007. doi: 10.1016/j.ejor.2005.03.061. URL http://dx.doi.org/10.1016/j.ejor.2005.03.061.

Rodríguez, B., Molina, J., Pérez, F., and Caballero, R. Interactive design of personalised tourism routes. *Tourism Management*, 33(4):926 – 940, 2012. ISSN 0261-5177. doi: http://dx.doi.org/10.1016/j.tourman.2011.09.014. URL http://www.sciencedirect.com/science/article/pii/S0261517711001993.

Schilde, M., Doerner, K. F., Hartl, R. F., and Kiechle, G. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, 2009. doi: 10.1007/s11721-009-0029-5. URL http://dx.doi.org/10.1007/s11721-009-0029-5.

Schulz, C., Hasle, G., Brodtkorb, A., and Hagen, T. GPU computing in discrete optimization. part ii: Survey focused on routing problems. *EURO Journal on Transportation and Logistics*, 2(1-2):159–186, 2013. ISSN 2192-

4376. doi: 10.1007/s13676-013-0026-0. URL `http://dx.doi.org/10.1007/s13676-013-0026-0`.

Silberholz, J. and Golden, B. L. The effective application of a new approach to the generalized orienteering problem. *J. Heuristics*, 16(3):393–415, 2010. doi: 10.1007/s10732-009-9104-8. URL `http://dx.doi.org/10.1007/s10732-009-9104-8`.

Solomon, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987. doi: 10.1287/opre.35.2.254. URL `http://dx.doi.org/10.1287/opre.35.2.254`.

Souffriau, W. *Automated Tourist Decision Support*. PhD thesis, Centre for Industrial Management, Katholieke Universiteit Leuven, Belgium, 2010.

Souffriau, W., Vansteenwegen, P., Berghe, G. V., and Van Oudheusden, D. A greedy randomised adaptive search procedure for the team orienteering problem. In *Proceedings of EU/MEeting 2008, Metaheuristics for Logistics and Vehicle Routing*, pages 1–9, 2008.

Souffriau, W., Vansteenwegen, P., Berghe, G. V., and Oudheusden, D. V. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1):53–63, 2013. doi: 10.1287/trsc.1110.0377. URL `http://dx.doi.org/10.1287/trsc.1110.0377`.

Sylejmani, K., Dorn, J., and Musliu, N. A tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning. In *12th International Conference on Hybrid Intelligent Systems, HIS 2012, Pune, India, December 4-7, 2012*, pages 300–305. IEEE, 2012. ISBN 978-1-4673-5114-0. doi: 10.1109/HIS.2012.6421351. URL `http://dx.doi.org/10.1109/HIS.2012.6421351`.

Tsiligirides, T. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, pages 797–809, 1984.

Vansteenwegen, P. and Oudheusden, D. V. The mobile tourist guide: An OR opportunity. *OR Insight*, 20(3):21–27, 2007. doi: 10.1057/ori.2007.17. URL http://dx.doi.org/10.1057/ori.2007.17.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Oudheusden, D. V. Iterated local search for the team orienteering problem with time windows. *Computers & OR*, 36(12):3281–3290, 2009. doi: 10.1016/j.cor.2009.03.008. URL http://dx.doi.org/10.1016/j.cor.2009.03.008.

Vansteenwegen, P., Souffriau, W., and Oudheusden, D. V. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011. doi: 10.1016/j.ejor.2010.03.045. URL http://dx.doi.org/10.1016/j.ejor.2010.03.045.

Verbeeck, C., Sörensen, K., Aghezzaf, E., and Vansteenwegen, P. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2):419–432, 2014. doi: 10.1016/j.ejor.2013.11.038. URL http://dx.doi.org/10.1016/j.ejor.2013.11.038.

Wang, X., Golden, B. L., and Wasil, E. A. Using a genetic algorithm to solve the generalized orienteering problem. In Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 263–274. Springer US, 2008. ISBN 978-0-387-77777-1. doi: 10.1007/978-0-387-77778-8_12. URL http://dx.doi.org/10.1007/978-0-387-77778-8_12.

Yu, J., Aslam, J. A., Karaman, S., and Rus, D. Optimal tourist problem and anytime planning of trip itineraries. *CoRR*, abs/1409.8536, 2014. URL http://arxiv.org/abs/1409.8536.

Appendix A

Detailed gap results

A.1 Orienteering Problem

The following tables give the gap to optimum result for the cooperative GRASP-ILS hybrid (`GRILS-T`) over all of the tested OP instances. The datasets `tsi32`, `tsi21` and `tsi33` are the 32-vertex, 21-vertex and 33-vertex datasets, respectively, of Tsiligirides (1984). The datasets `chao66` and `chao64` are the 66-vertex and 64-vertex datasets of Chao et al. (1996b). The rest of the datasets were converted from TSPLIB problems by Fischetti et al. (1998). The full set of problem instances with pre-generated distance tables is available online [1].

d_{lim} is the distance budget for the given instance. *Optimum* is determined as described in Section 3.1.1. *Min.*, *Avg.* and *Max.* give the minimum, arithmetic mean and maximum gap for the given instance over 10 test runs. The gap was calculated as $\left(1 - \frac{S}{S_{opt}}\right) 100\%$ where S_{opt} is the optimum for the instance and S is the score of the test run.

The gap results are given for test runs with 23 parallel workers. The workload was partitioned by assigning a fixed number of 2000 iterations to each worker.

[1] `http://josilber.scripts.mit.edu/gop.zip`

Table A.1: Gap results for cooperative GRASP-ILS hybrid: Tsiligirides instances

Problem	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
tsi32	5	10	0.00%	0.00%	0.00%
tsi32	10	15	0.00%	0.00%	0.00%
tsi32	15	45	0.00%	0.00%	0.00%
tsi32	20	65	0.00%	0.00%	0.00%
tsi32	25	90	0.00%	0.00%	0.00%
tsi32	30	110	0.00%	0.00%	0.00%
tsi32	35	135	0.00%	0.00%	0.00%
tsi32	40	155	0.00%	0.00%	0.00%
tsi32	46	175	0.00%	0.00%	0.00%
tsi32	50	190	0.00%	0.00%	0.00%
tsi32	55	205	0.00%	0.00%	0.00%
tsi32	60	225	0.00%	0.22%	2.22%
tsi32	65	240	0.00%	0.00%	0.00%
tsi32	70	260	0.00%	0.00%	0.00%
tsi32	73	265	0.00%	0.00%	0.00%
tsi32	75	270	0.00%	0.00%	0.00%
tsi32	80	280	0.00%	0.00%	0.00%
tsi32	85	285	0.00%	0.00%	0.00%
tsi21	15	120	0.00%	0.00%	0.00%
tsi21	20	200	0.00%	0.00%	0.00%
tsi21	23	210	0.00%	0.00%	0.00%
tsi21	25	230	0.00%	0.00%	0.00%
tsi21	27	230	0.00%	0.00%	0.00%
tsi21	30	265	0.00%	0.00%	0.00%
tsi21	32	300	0.00%	0.00%	0.00%
tsi21	35	320	0.00%	0.00%	0.00%
tsi21	38	360	0.00%	0.00%	0.00%
tsi21	40	395	0.00%	0.00%	0.00%
tsi21	45	450	0.00%	0.00%	0.00%
tsi33	15	170	0.00%	0.00%	0.00%
tsi33	20	200	0.00%	0.00%	0.00%
tsi33	25	260	0.00%	0.00%	0.00%
tsi33	30	320	0.00%	0.00%	0.00%
tsi33	35	390	0.00%	0.00%	0.00%
tsi33	40	430	0.00%	0.00%	0.00%

Table A.1: Continued

Problem	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
tsi33	45	470	0.00%	0.00%	0.00%
tsi33	50	520	0.00%	0.00%	0.00%
tsi33	55	550	0.00%	0.00%	0.00%
tsi33	60	580	0.00%	0.00%	0.00%
tsi33	65	610	0.00%	0.00%	0.00%
tsi33	70	640	0.00%	0.00%	0.00%
tsi33	75	670	0.00%	0.00%	0.00%
tsi33	80	710	0.00%	0.00%	0.00%
tsi33	85	740	0.00%	0.00%	0.00%
tsi33	90	770	0.00%	0.00%	0.00%
tsi33	95	790	0.00%	0.00%	0.00%
tsi33	100	800	0.00%	0.00%	0.00%
tsi33	105	800	0.00%	0.00%	0.00%
tsi33	110	800	0.00%	0.00%	0.00%

Table A.2: Gap results for cooperative GRASP-ILS hybrid: Chao et al. instances

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
chao64	5	10	0.00%	0.00%	0.00%
chao64	10	40	0.00%	0.00%	0.00%
chao64	15	120	0.00%	0.00%	0.00%
chao64	20	205	0.00%	0.00%	0.00%
chao64	25	290	0.00%	0.00%	0.00%
chao64	30	400	0.00%	0.00%	0.00%
chao64	35	465	0.00%	0.00%	0.00%
chao64	40	575	0.00%	0.35%	3.48%
chao64	45	650	0.00%	1.54%	3.08%
chao64	50	730	0.00%	0.96%	2.74%
chao64	55	825	0.00%	2.06%	3.64%
chao64	60	915	1.09%	2.73%	4.37%
chao64	65	980	0.00%	1.53%	3.06%
chao64	70	1070	0.00%	1.87%	4.67%
chao64	75	1140	0.00%	1.23%	3.07%
chao64	80	1215	0.00%	1.23%	1.65%
chao64	85	1270	0.00%	0.00%	0.00%

Table A.2: Continued

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
chao64	90	1340	0.00%	0.37%	0.75%
chao64	95	1395	0.00%	0.00%	0.00%
chao64	100	1465	0.00%	0.00%	0.00%
chao64	105	1520	0.00%	0.00%	0.00%
chao64	110	1560	0.00%	0.00%	0.00%
chao64	115	1595	0.00%	0.00%	0.00%
chao64	120	1635	0.00%	0.00%	0.00%
chao64	125	1670	0.00%	0.00%	0.00%
chao64	130	1680	0.00%	0.00%	0.00%
chao66	15	96	0.00%	0.00%	0.00%
chao66	20	294	0.00%	0.00%	0.00%
chao66	25	390	0.00%	0.15%	1.54%
chao66	30	474	1.27%	1.27%	1.27%
chao66	35	576	0.00%	1.67%	3.12%
chao66	40	714	0.00%	0.00%	0.00%
chao66	45	816	0.00%	0.74%	1.47%
chao66	50	900	0.00%	0.87%	2.00%
chao66	55	984	0.00%	0.49%	0.61%
chao66	60	1062	0.00%	0.23%	1.13%
chao66	65	1116	0.00%	0.00%	0.00%
chao66	70	1188	0.00%	0.00%	0.00%
chao66	75	1236	0.00%	0.00%	0.00%
chao66	80	1284	0.00%	0.00%	0.00%

Table A.3: Gap results for cooperative GRASP-ILS hybrid: Fischetti et al. instances

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
		Generation 1			
att48	5314	31	0.00%	0.00%	0.00%
gr48	2523	31	0.00%	0.00%	0.00%
hk48	5731	30	0.00%	0.00%	0.00%
eil51	213	29	0.00%	0.00%	0.00%
brazil58	12698	46	0.00%	1.09%	2.17%
st70	338	43	0.00%	0.47%	2.33%
eil76	269	47	0.00%	1.28%	2.13%

Table A.3: Continued

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
pr76	54080	49	0.00%	1.84%	2.04%
gr96	27605	64	3.12%	5.31%	6.25%
rat99	606	52	0.00%	1.35%	1.92%
kroA100	10641	56	0.00%	0.00%	0.00%
kroB100	11071	58	0.00%	0.00%	0.00%
kroC100	10375	56	5.36%	5.71%	7.14%
kroD100	10647	59	1.69%	2.20%	3.39%
kroE100	11034	57	0.00%	0.00%	0.00%
rd100	3955	61	0.00%	1.80%	3.28%
eil101	315	64	0.00%	0.62%	1.56%
lin105	7190	66	0.00%	0.00%	0.00%
pr107	22152	54	0.00%	0.00%	0.00%
gr120	3471	75	1.33%	4.40%	6.67%
pr124	29515	75	0.00%	0.00%	0.00%
bier127	59141	103	0.00%	0.00%	0.00%
pr136	48386	71	4.23%	4.23%	4.23%
gr137	34927	81	0.00%	0.99%	1.23%
pr144	29269	77	0.00%	0.52%	3.90%
kroA150	13262	86	2.33%	4.53%	5.81%
kroB150	13065	87	2.30%	5.29%	8.05%
pr152	36841	77	1.30%	4.16%	6.49%
u159	21040	93	0.00%	0.65%	1.08%
rat195	1162	102	1.96%	2.94%	3.92%
d198	7890	123	3.25%	6.10%	7.32%
kroA200	14684	117	3.42%	4.27%	5.98%
kroB200	14719	119	5.04%	7.06%	9.24%
gr202	20080	147	2.72%	3.88%	4.76%
ts225	63322	125	2.40%	2.88%	4.80%
pr226	40185	134	9.70%	15.75%	17.91%
gr229	67301	176	1.70%	1.82%	2.27%
gil262	1189	158	4.43%	8.99%	12.03%
pr264	24568	132	0.00%	0.00%	0.00%
pr299	24096	162	4.32%	4.75%	5.56%
lin318	21045	205	10.24%	11.66%	14.63%
rd400	7641	239	6.69%	8.28%	9.62%
Generation 2					
att48	5314	1717	0.00%	0.00%	0.00%

Table A.3: Continued

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
gr48	2523	1761	0.28%	0.59%	0.62%
hk48	5731	1614	0.00%	0.62%	1.86%
eil51	213	1674	0.00%	0.22%	0.72%
brazil58	12698	2220	0.00%	0.16%	0.81%
st70	338	2286	0.04%	0.16%	0.44%
eil76	269	2550	0.63%	2.09%	3.14%
pr76	54080	2708	0.00%	0.42%	0.89%
gr96	27605	3425	3.27%	3.74%	4.41%
rat99	606	2944	1.15%	2.26%	4.62%
kroA100	10641	3212	0.00%	0.00%	0.00%
kroB100	11071	3241	0.74%	1.23%	1.85%
kroC100	10375	2947	0.00%	0.24%	1.36%
kroD100	10647	3307	0.24%	1.93%	3.87%
kroE100	11034	3090	0.06%	0.55%	1.65%
rd100	3955	3359	0.00%	0.68%	0.80%
eil101	315	3655	0.55%	0.84%	1.20%
lin105	7190	3544	0.23%	0.30%	0.48%
pr107	22152	2667	0.00%	0.00%	0.00%
gr120	3471	4371	0.96%	3.88%	7.62%
pr124	29515	3917	0.00%	0.05%	0.46%
bier127	59141	5383	0.85%	1.28%	2.02%
pr136	48386	4309	2.88%	5.07%	6.20%
gr137	34927	4294	0.37%	0.47%	0.63%
pr144	29269	4003	0.00%	0.29%	1.00%
kroA150	13262	4918	0.71%	2.89%	3.76%
kroB150	13065	4869	1.66%	5.17%	7.23%
pr152	36841	4279	0.91%	3.00%	5.63%
u159	21040	4960	1.15%	1.63%	3.19%
rat195	1162	5791	3.26%	4.57%	6.20%
d198	7890	6670	0.99%	2.57%	4.74%
kroA200	14684	6547	3.42%	5.59%	8.14%
kroB200	14719	6419	2.26%	3.94%	6.15%
gr202	20080	7848	2.45%	3.38%	4.36%
ts225	63322	6834	2.66%	2.89%	4.01%
pr226	40185	6615	7.32%	7.81%	8.06%
gr229	67301	9187	0.36%	1.19%	2.04%
gil262	1189	8321	2.92%	5.45%	8.11%

Table A.3: Continued

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
pr264	24568	6654	0.00%	0.00%	0.00%
pr299	24096	9161	2.61%	6.61%	9.29%
lin318	21045	10900	4.04%	6.93%	10.56%
rd400	7641	13648	7.52%	9.07%	10.97%
		Generation 3			
att48	5314	1049	0.00%	0.00%	0.00%
gr48	2523	1480	0.00%	0.00%	0.00%
hk48	5731	1764	0.00%	0.00%	0.00%
eil51	213	1399	0.00%	0.06%	0.14%
brazil58	12698	1702	0.00%	0.04%	0.35%
st70	338	2108	0.00%	0.77%	1.80%
eil76	269	2467	0.20%	0.99%	1.95%
pr76	54080	2430	0.00%	0.13%	0.16%
gr96	27605	3182	1.16%	5.08%	7.95%
rat99	606	2908	1.34%	2.50%	3.37%
kroA100	10641	3211	0.72%	0.82%	1.03%
kroB100	11071	2804	0.00%	0.93%	1.78%
kroC100	10375	3155	4.25%	4.25%	4.25%
kroD100	10647	3167	0.51%	1.31%	2.12%
kroE100	11034	3049	1.90%	3.10%	3.87%
rd100	3955	2926	0.07%	3.66%	5.09%
eil101	315	3345	1.55%	1.93%	2.69%
lin105	7190	2986	3.08%	3.08%	3.08%
pr107	22152	1877	6.45%	6.45%	6.45%
gr120	3471	3779	2.38%	5.41%	9.69%
pr124	29515	3557	0.22%	0.22%	0.22%
bier127	59141	2365	0.72%	2.07%	2.62%
pr136	48386	4390	3.28%	4.25%	5.51%
gr137	34927	3979	1.01%	7.45%	12.49%
pr144	29269	3809	6.33%	7.83%	9.00%
kroA150	13262	5039	0.14%	0.36%	0.67%
kroB150	13065	5314	1.39%	4.17%	7.43%
pr152	36841	3905	0.00%	0.04%	0.28%
u159	21040	5272	2.26%	2.26%	2.33%
rat195	1162	6195	2.34%	3.88%	4.94%
d198	7890	6320	2.88%	4.25%	5.82%
kroA200	14684	6123	2.94%	4.51%	6.70%

Table A.3: Continued

Dataset	d_{lim}	*Optimum*	*Min.*	*Avg.*	*Max.*
kroB200	14719	6266	0.45%	1.89%	4.12%
gr202	20080	8632	0.66%	2.48%	3.49%
ts225	63322	7575	6.44%	7.14%	9.60%
pr226	40185	6993	5.62%	6.97%	12.21%
gr229	67301	6347	2.39%	3.00%	4.35%
gil262	1189	9246	3.21%	5.55%	6.98%
pr264	24568	8137	1.18%	4.71%	7.02%
pr299	24096	10358	2.09%	4.84%	7.34%
lin318	21045	10382	6.24%	11.00%	12.68%
rd400	7641	13229	5.27%	5.92%	7.43%

A.2 Generalized Orienteering Problem

Table A.4 gives the gap to the best known result for the cooperative GRASP-ILS hybrid (`GRILS-T`) over the 27-vertex dataset that is reproduced, among others, in (Wang et al. 2008). The dataset is also distributed online [2] with pre-generated distance tables. In each instance, the distance limit d_{lim} was set to 5000.

In Table A.4, *k* is the parameter k in the objective function. *Wt* refers to the weight vector used. For $Wt = 0$, the value of each attribute is 0.25. For $Wt > 0$, the attribute with the given number is set to 1.0 and the rest of the attributes are set to 0. *Best known* is the result of the `2PIA` algorithm as given in (Silberholz and Golden 2010). *Min.*, *Avg.* and *Max.* give the minimum, arithmetic mean and maximum gap for the given instance over 10 test runs. The gap was calculated as $\left(1 - \frac{S}{S_{best}}\right) 100\%$ where S_{best} is the best known result for the instance and S is the score of the test run. Following the convention of the earlier publications using this benchmark, we round the results to 2 fractional digits.

Test runs were performed with 23 parallel workers. The workload was partitioned by assigning a fixed number of 2000 iterations to each worker.

[2] `http://josilber.scripts.mit.edu/gop.zip`

Table A.4: Gap results for cooperative GRASP-ILS hybrid: the ”27 Chinese Cities” benchmark

k	*Wt*	*Best known*	*Min.*	*Avg.*	*Max.*
1	0	99.50	0.00%	0.00%	0.00%
1	1	105.00	0.00%	0.00%	0.00%
1	2	97.00	0.00%	0.00%	0.00%
1	3	102.00	0.00%	0.00%	0.00%
1	4	96.00	0.00%	0.00%	0.00%
3	0	16.76	0.00%	0.00%	0.00%
3	1	17.95	0.00%	0.65%	1.74%
3	2	17.04	0.00%	0.00%	0.00%
3	3	17.45	0.00%	0.00%	0.00%
3	4	16.78	2.18%	3.55%	4.56%
4	0	13.71	0.00%	0.23%	0.25%
4	1	14.69	0.28%	0.79%	1.46%
4	2	13.99	0.00%	0.00%	0.00%
4	3	14.29	0.00%	0.00%	0.00%
4	4	13.84	0.00%	3.52%	6.05%
5	0	12.38	0.60%	0.66%	1.22%
5	1	13.10	0.39%	0.79%	1.74%
5	2	12.56	0.00%	0.14%	0.23%
5	3	12.78	0.00%	0.00%	0.00%
5	4	12.43	1.05%	3.19%	6.90%
10	0	10.54	0.56%	0.62%	0.62%
10	1	10.75	0.10%	0.21%	0.63%
10	2	10.57	0.09%	0.10%	0.12%
10	3	10.62	0.02%	0.02%	0.02%
10	4	10.48	0.20%	1.54%	9.28%

A.3 Team Orienteering Problem with Time Windows

The TOPTW benchmark [3] was originally designed by Montemanni and Gambardella (2009), from the vehicle routing problem datasets of Solomon (1987) and Cordeau et al. (1997). Each instance of the dataset can be solved as a TOPTW problem with the number of tours $1 \leq m \leq 4$.

[3] `http://www.mech.kuleuven.be/en/cib/op/`

Tables A.5–A.8 provide the results of `GRILS-T` for the instances based on the dataset of Solomon. Tables A.9–A.12 list the results for the instances derived from the dataset of Cordeau et al.

d_{lim} is the distance budget per tour for the given instance. *Best known* is the reference best solution as given in (Cura 2014). There are some inconsistencies in reporting the best known solutions for TOPTW instances, for example Vansteenwegen et al. (2009) give a higher best known score in some instances even though its publication predates the work of Cura by several years.

In cases where `GRILS-T` found a new best solution, we've additionally verified that is is higher than the best solutions reported in (Vansteenwegen et al. 2009) and (Labadi et al. 2011). These results are printed in bold.

Min., *Avg.* and *Max.* give the minimum, arithmetic mean and maximum gap for the given instance over 10 test runs. The gap was calculated as $\left(1 - \frac{S}{S_{best}}\right) 100\%$ where S_{best} is the best known solution score for the instance and S is the score of the test run.

Test runs were performed with 23 parallel workers. The workload was partitioned by assigning a fixed number of 2000 iterations to each worker.

Table A.5: Gap results for cooperative GRASP-ILS hybrid: the Solomon instances ($m = 1$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	320.00	0.00%	0.00%	0.00%
c102	1236	360.00	0.00%	0.00%	0.00%
c103	1236	400.00	0.00%	0.00%	0.00%
c104	1236	420.00	0.00%	0.00%	0.00%
c105	1236	340.00	0.00%	0.00%	0.00%
c106	1236	340.00	0.00%	0.00%	0.00%
c107	1236	370.00	0.00%	0.00%	0.00%
c108	1236	370.00	0.00%	0.00%	0.00%
c109	1236	380.00	0.00%	0.00%	0.00%
r101	230	198.00	0.00%	0.00%	0.00%
r102	230	286.00	0.00%	0.00%	0.00%
r103	230	293.00	0.00%	0.00%	0.00%
r104	230	303.00	0.00%	0.00%	0.00%

Table A.5: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
r105	230	247.00	0.00%	0.00%	0.00%
r106	230	293.00	0.00%	0.00%	0.00%
r107	230	299.00	0.67%	0.70%	1.00%
r108	230	308.00	0.00%	0.00%	0.00%
r109	230	277.00	1.08%	1.08%	1.08%
r110	230	284.00	1.06%	1.06%	1.06%
r111	230	297.00	0.00%	0.00%	0.00%
r112	230	298.00	0.00%	0.17%	0.67%
rc101	240	219.00	1.37%	1.37%	1.37%
rc102	240	266.00	0.00%	0.00%	0.00%
rc103	240	266.00	0.00%	0.00%	0.00%
rc104	240	301.00	0.00%	0.00%	0.00%
rc105	240	244.00	0.00%	0.00%	0.00%
rc106	240	252.00	0.00%	0.00%	0.00%
rc107	240	277.00	0.00%	0.11%	1.08%
rc108	240	298.00	0.00%	0.00%	0.00%
c201	3390	870.00	0.00%	0.00%	0.00%
c202	3390	930.00	0.00%	0.43%	1.08%
c203	3390	960.00	0.00%	0.10%	1.04%
c205	3390	910.00	0.00%	0.00%	0.00%
c206	3390	930.00	0.00%	0.54%	1.08%
c207	3390	930.00	0.00%	0.00%	0.00%
c208	3390	950.00	0.00%	0.42%	1.05%
r201	1000	796.70	1.72%	2.11%	2.72%
r202	1000	930.00	2.69%	3.58%	4.62%
r203	1000	1020.00	0.98%	2.26%	3.14%
r204	1000	1076.30	-0.34%	0.24%	0.86%
r205	1000	953.00	1.15%	1.97%	3.46%
r206	1000	1023.60	0.16%	1.52%	3.18%
r207	1000	1069.00	0.00%	1.32%	2.43%
r208	1000	1101.50	-0.50%	0.80%	1.68%
r209	1000	948.00	1.58%	2.68%	2.95%
r210	1000	982.00	-0.20%	1.68%	2.95%
r211	1000	1041.00	0.96%	2.07%	3.94%
rc201	960	795.00	1.13%	1.74%	2.26%
rc202	960	930.00	-0.32%	0.96%	1.83%
rc203	960	988.20	0.43%	0.95%	1.54%

Table A.5: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
rc204	960	1140.00	0.35%	1.41%	3.25%
rc205	960	854.00	0.12%	0.71%	1.29%
rc206	960	890.20	-0.54%	1.33%	2.38%
rc207	960	977.00	0.00%	1.73%	4.09%
rc208	960	1043.50	**-0.14%**	1.16%	3.31%

Table A.6: Gap results for cooperative GRASP-ILS hybrid: the Solomon instances ($m = 2$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	590.00	1.69%	1.69%	1.69%
c102	1236	660.00	1.52%	1.52%	1.52%
c103	1236	720.00	1.39%	1.39%	1.39%
c104	1236	760.00	0.00%	0.00%	0.00%
c105	1236	640.00	0.00%	0.00%	0.00%
c106	1236	620.00	0.00%	0.00%	0.00%
c107	1236	670.00	1.49%	1.49%	1.49%
c108	1236	680.00	0.00%	0.00%	0.00%
c109	1236	720.00	1.39%	1.39%	1.39%
r101	230	349.00	0.00%	0.00%	0.00%
r102	230	508.00	0.00%	0.12%	0.79%
r103	230	519.00	-0.19%	0.67%	1.35%
r104	230	549.00	**-0.18%**	0.29%	0.91%
r105	230	453.00	2.21%	2.21%	2.21%
r106	230	529.00	0.00%	1.08%	1.89%
r107	230	533.00	**-0.94%**	0.34%	2.06%
r108	230	558.00	0.00%	0.50%	1.08%
r109	230	506.00	0.99%	1.34%	1.78%
r110	230	525.00	2.29%	2.63%	3.24%
r111	230	544.00	0.55%	1.19%	2.02%
r112	230	544.00	0.00%	1.47%	3.12%
rc101	240	427.00	0.70%	0.70%	0.70%
rc102	240	505.00	0.20%	0.26%	0.40%
rc103	240	523.00	0.00%	0.48%	2.10%
rc104	240	575.00	0.17%	0.90%	2.61%
rc105	240	480.00	0.00%	0.42%	0.83%

Table A.6: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
rc106	240	483.00	0.21%	0.21%	0.21%
rc107	240	531.40	1.02%	1.37%	2.15%
rc108	240	554.00	0.18%	1.48%	2.89%
c201	3390	1452.00	0.83%	1.31%	1.52%
c202	3390	1470.00	0.68%	1.43%	2.04%
c203	3390	1472.00	1.49%	1.49%	1.49%
c205	3390	1470.00	0.68%	1.22%	1.36%
c206	3390	1480.00	0.68%	1.15%	1.35%
c207	3390	1484.00	0.94%	0.94%	0.94%
c208	3390	1486.00	1.08%	1.08%	1.08%
r201	1000	1242.00	1.61%	2.46%	3.38%
r202	1000	1344.00	1.49%	3.39%	4.54%
r203	1000	1416.00	2.47%	3.25%	3.74%
r204	1000	1458.00	1.58%	1.87%	2.26%
r205	1000	1380.00	1.67%	2.93%	3.77%
r206	1000	1430.00	0.91%	1.87%	3.22%
r207	1000	1458.00	1.30%	1.60%	2.13%
r208	1000	1458.00	1.17%	1.17%	1.17%
r209	1000	1404.00	2.85%	3.67%	4.49%
r210	1000	1415.00	2.33%	3.24%	4.17%
r211	1000	1457.00	1.65%	2.13%	2.54%
rc201	960	1377.00	0.73%	2.47%	3.27%
rc202	960	1502.40	1.49%	2.65%	4.82%
rc203	960	1627.00	2.77%	3.79%	4.92%
rc204	960	1710.20	0.36%	1.51%	2.41%
rc205	960	1458.00	1.30%	3.15%	5.08%
rc206	960	1528.00	0.65%	2.57%	3.99%
rc207	960	1582.00	-0.32%	1.19%	2.59%
rc208	960	1676.10	0.18%	1.55%	2.51%

Table A.7: Gap results for cooperative GRASP-ILS hybrid: the Solomon instances ($m = 3$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	810.00	1.23%	1.23%	1.23%
c102	1236	920.00	1.09%	1.41%	2.17%

Table A.7: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c103	1236	990.00	1.01%	2.63%	3.03%
c104	1236	1030.00	0.00%	1.17%	1.94%
c105	1236	870.00	1.15%	1.61%	2.30%
c106	1236	870.00	0.00%	1.38%	2.30%
c107	1236	910.00	1.10%	1.10%	1.10%
c108	1236	920.00	1.09%	1.09%	1.09%
c109	1236	970.00	1.03%	1.65%	2.06%
r101	230	484.00	0.62%	0.97%	2.27%
r102	230	694.00	0.86%	2.10%	3.31%
r103	230	746.00	1.47%	2.36%	2.95%
r104	230	777.00	0.13%	1.13%	2.06%
r105	230	619.30	1.82%	2.62%	3.44%
r106	230	729.00	1.51%	2.77%	4.12%
r107	230	760.00	0.26%	1.11%	1.84%
r108	230	797.00	0.88%	1.47%	2.63%
r109	230	710.00	1.97%	2.82%	3.80%
r110	230	736.00	2.31%	3.94%	4.76%
r111	230	773.00	0.13%	0.62%	2.07%
r112	230	776.00	0.90%	1.56%	2.84%
rc101	240	621.00	0.00%	0.89%	1.61%
rc102	240	714.00	0.70%	1.67%	3.08%
rc103	240	764.00	0.13%	2.74%	3.80%
rc104	240	834.00	0.24%	0.89%	1.80%
rc105	240	682.00	0.29%	1.52%	2.35%
rc106	240	706.00	0.14%	1.20%	2.69%
rc107	240	773.00	0.91%	1.84%	2.85%
rc108	240	789.00	0.51%	1.15%	2.53%
c201	3390	1810.00	1.10%	1.93%	2.76%
c202	3390	1810.00	1.10%	1.60%	2.21%
c203	3390	1810.00	1.66%	1.88%	2.21%
c205	3390	1810.00	1.10%	1.16%	1.66%
c206	3390	1810.00	1.10%	1.10%	1.10%
c207	3390	1810.00	1.10%	1.10%	1.10%
c208	3390	1810.00	1.10%	1.10%	1.10%
r201	1000	1438.40	1.84%	2.43%	2.95%
r202	1000	1458.00	1.37%	1.72%	2.06%
r203	1000	1458.00	1.17%	1.17%	1.17%

Table A.7: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
r204	1000	1458.00	1.17%	1.17%	1.17%
r205	1000	1458.00	1.17%	1.17%	1.17%
r206	1000	1458.00	1.17%	1.17%	1.17%
r207	1000	1458.00	1.17%	1.17%	1.17%
r208	1000	1458.00	1.17%	1.17%	1.17%
r209	1000	1458.00	1.17%	1.17%	1.17%
r210	1000	1458.00	1.17%	1.17%	1.17%
r211	1000	1458.00	1.17%	1.17%	1.17%
rc201	960	1689.40	1.21%	1.64%	2.10%
rc202	960	1724.00	0.75%	1.16%	1.39%
rc203	960	1724.00	0.17%	0.17%	0.17%
rc204	960	1724.00	0.17%	0.17%	0.17%
rc205	960	1719.00	1.22%	1.48%	1.98%
rc206	960	1724.00	0.17%	0.17%	0.17%
rc207	960	1724.00	0.17%	0.17%	0.17%
rc208	960	1724.00	0.17%	0.17%	0.17%

Table A.8: Gap results for cooperative GRASP-ILS hybrid: the Solomon instances ($m = 4$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	1020.00	0.98%	0.98%	0.98%
c102	1236	1150.00	1.74%	2.09%	2.61%
c103	1236	1210.00	1.65%	2.98%	4.13%
c104	1236	1260.00	1.59%	2.06%	2.38%
c105	1236	1060.00	0.00%	0.85%	1.89%
c106	1236	1080.00	1.85%	2.96%	3.70%
c107	1236	1120.00	0.89%	2.14%	2.68%
c108	1236	1130.00	0.88%	1.95%	2.65%
c109	1236	1190.00	1.68%	2.02%	3.36%
r101	230	611.00	0.49%	1.42%	2.45%
r102	230	843.00	2.02%	4.09%	5.81%
r103	230	928.00	1.40%	2.53%	3.66%
r104	230	969.00	0.83%	1.71%	2.58%
r105	230	778.00	2.70%	3.87%	4.88%
r106	230	906.00	1.43%	3.27%	5.08%

Table A.8: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
r107	230	950.00	1.16%	2.27%	4.00%
r108	230	994.00	1.11%	1.85%	3.32%
r109	230	885.00	1.02%	3.72%	5.08%
r110	230	915.00	4.15%	5.37%	6.99%
r111	230	952.00	1.37%	2.14%	3.36%
r112	230	967.00	1.14%	3.25%	4.76%
rc101	240	808.00	0.00%	1.49%	2.72%
rc102	240	902.00	0.44%	1.37%	2.55%
rc103	240	974.00	1.75%	2.95%	3.90%
rc104	240	1064.00	0.94%	1.11%	1.69%
rc105	240	875.00	2.06%	2.82%	4.34%
rc106	240	909.00	1.21%	2.08%	2.97%
rc107	240	980.00	0.00%	1.03%	2.04%
rc108	240	1023.00	0.98%	2.06%	2.74%
c201	3390	1810.00	1.10%	1.10%	1.10%
c202	3390	1810.00	1.10%	1.10%	1.10%
c203	3390	1810.00	1.10%	1.10%	1.10%
c205	3390	1810.00	1.10%	1.10%	1.10%
c206	3390	1810.00	1.10%	1.10%	1.10%
c207	3390	1810.00	1.10%	1.10%	1.10%
c208	3390	1810.00	1.10%	1.10%	1.10%
r201	1000	1458.00	1.17%	1.17%	1.17%
r202	1000	1458.00	1.17%	1.17%	1.17%
r203	1000	1458.00	1.17%	1.17%	1.17%
r204	1000	1458.00	1.17%	1.17%	1.17%
r205	1000	1458.00	1.17%	1.17%	1.17%
r206	1000	1458.00	1.17%	1.17%	1.17%
r207	1000	1458.00	1.17%	1.17%	1.17%
r208	1000	1458.00	1.17%	1.17%	1.17%
r209	1000	1458.00	1.17%	1.17%	1.17%
r210	1000	1458.00	1.17%	1.17%	1.17%
r211	1000	1458.00	1.17%	1.17%	1.17%
rc201	960	1724.00	0.17%	0.17%	0.17%
rc202	960	1724.00	0.17%	0.17%	0.17%
rc203	960	1724.00	0.17%	0.17%	0.17%
rc204	960	1724.00	0.17%	0.17%	0.17%
rc205	960	1724.00	0.17%	0.17%	0.17%

Table A.8: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
rc206	960	1724.00	0.17%	0.17%	0.17%
rc207	960	1724.00	0.17%	0.17%	0.17%
rc208	960	1724.00	0.17%	0.17%	0.17%

Table A.9: Gap results for cooperative GRASP-ILS hybrid: the Cordeau et al. instances ($m = 1$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr01	1000	308.00	0.00%	0.00%	0.00%
pr02	1000	404.00	0.00%	0.42%	0.99%
pr03	1000	394.00	0.00%	0.03%	0.25%
pr04	1000	489.00	2.25%	3.48%	4.91%
pr05	1000	594.00	1.68%	3.08%	4.21%
pr06	1000	590.00	1.69%	3.64%	6.44%
pr07	1000	298.00	1.68%	2.08%	2.35%
pr08	1000	463.00	1.94%	2.33%	2.38%
pr09	1000	490.00	-0.61%	1.84%	4.29%
pr10	1000	588.40	1.09%	2.04%	2.79%
pr11	1000	353.00	1.42%	1.93%	3.12%
pr12	1000	442.00	1.36%	1.54%	1.81%
pr13	1000	466.00	**-0.21%**	1.55%	4.08%
pr14	1000	560.10	2.16%	3.37%	5.73%
pr15	1000	707.00	1.13%	4.29%	6.22%
pr16	1000	652.60	1.62%	4.87%	7.45%
pr17	1000	362.00	1.10%	2.02%	2.76%
pr18	1000	539.00	2.04%	7.11%	10.39%
pr19	1000	551.60	1.74%	4.28%	5.55%
pr20	1000	656.60	3.90%	5.22%	6.18%

Table A.10: Gap results for cooperative GRASP-ILS hybrid: the Cordeau et al. instances ($m = 2$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr01	1000	502.00	2.99%	3.21%	4.38%
pr02	1000	714.00	2.24%	3.47%	4.76%

Table A.10: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr03	1000	741.00	0.94%	2.19%	3.91%
pr04	1000	917.00	0.11%	2.43%	5.67%
pr05	1000	1101.00	4.00%	5.40%	7.45%
pr06	1000	1070.20	0.86%	6.23%	9.64%
pr07	1000	566.00	1.77%	2.69%	3.00%
pr08	1000	826.20	2.57%	3.59%	4.74%
pr09	1000	883.40	**-1.54%**	2.78%	7.63%
pr10	1000	1117.00	0.63%	4.43%	8.68%
pr11	1000	566.00	1.41%	2.83%	4.06%
pr12	1000	768.00	2.73%	3.91%	4.95%
pr13	1000	832.00	1.92%	4.17%	6.25%
pr14	1000	999.00	1.90%	4.69%	7.11%
pr15	1000	1210.40	1.93%	5.43%	8.38%
pr16	1000	1217.80	6.88%	10.34%	12.63%
pr17	1000	652.00	2.76%	4.08%	5.37%
pr18	1000	937.00	3.84%	5.38%	7.68%
pr19	1000	1017.00	5.90%	8.34%	10.23%
pr20	1000	1224.40	5.18%	7.61%	10.16%

Table A.11: Gap results for cooperative GRASP-ILS hybrid: the Cordeau et al. instances ($m = 3$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr01	1000	622.00	2.89%	3.47%	4.66%
pr02	1000	939.00	1.60%	2.92%	4.05%
pr03	1000	1010.00	1.39%	3.41%	5.84%
pr04	1000	1286.00	3.50%	5.00%	7.23%
pr05	1000	1481.00	3.31%	5.54%	7.63%
pr06	1000	1501.00	3.66%	6.49%	9.39%
pr07	1000	742.00	2.43%	3.26%	4.31%
pr08	1000	1139.00	4.21%	5.72%	6.58%
pr09	1000	1272.00	3.85%	7.19%	9.91%
pr10	1000	1567.00	2.36%	5.40%	7.66%
pr11	1000	654.00	2.29%	2.63%	3.06%
pr12	1000	997.00	2.91%	3.69%	4.61%
pr13	1000	1145.00	3.49%	6.33%	7.95%

Table A.11: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr14	1000	1357.40	4.97%	6.62%	8.58%
pr15	1000	1654.00	2.78%	6.59%	9.01%
pr16	1000	1654.60	7.11%	9.04%	11.34%
pr17	1000	841.00	3.09%	4.46%	5.95%
pr18	1000	1276.00	3.37%	8.31%	11.52%
pr19	1000	1403.00	5.13%	8.98%	11.26%
pr20	1000	1677.60	5.40%	7.91%	9.93%

Table A.12: Gap results for cooperative GRASP-ILS hybrid: the Cordeau et al. instances ($m = 4$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr01	1000	657.00	1.52%	1.70%	1.98%
pr02	1000	1073.00	2.89%	4.21%	4.85%
pr03	1000	1232.00	3.25%	5.30%	6.41%
pr04	1000	1585.00	3.79%	6.35%	7.95%
pr05	1000	1838.00	5.71%	7.54%	8.76%
pr06	1000	1840.40	1.38%	4.16%	6.76%
pr07	1000	872.00	3.67%	4.99%	6.08%
pr08	1000	1377.00	4.21%	6.08%	7.63%
pr09	1000	1604.00	6.23%	8.17%	9.73%
pr10	1000	1943.00	5.10%	7.03%	9.21%
pr11	1000	657.00	1.52%	1.52%	1.52%
pr12	1000	1130.10	3.64%	4.41%	5.50%
pr13	1000	1386.00	3.90%	6.53%	8.08%
pr14	1000	1651.00	2.79%	3.86%	5.69%
pr15	1000	2065.00	5.18%	7.74%	9.64%
pr16	1000	2017.00	6.54%	8.03%	9.62%
pr17	1000	934.00	3.32%	4.70%	5.46%
pr18	1000	1539.00	6.37%	9.38%	11.44%
pr19	1000	1750.00	7.20%	9.85%	12.11%
pr20	1000	2062.00	5.77%	7.42%	8.73%

A.4 Multi-Constraint Team Orienteering Problem with Multiple Time Windows

We used the dataset with multiple time windows designed by Souffriau et al. (2013). This dataset is based on a subset of the vehicle routing problems of Solomon (1987) and Cordeau et al. (1997).

The results of `GRILS-T` by number of tours m are given in Tables A.13–A.16. d_{lim} is the distance budget per tour for the given instance. *Best known* is taken from the online accompanying materials [4] of (Souffriau et al. 2013).

Min., *Avg.* and *Max.* give the minimum, arithmetic mean and maximum gap for the given instance over 10 test runs. The gap was calculated as $\left(1 - \frac{S}{S_{best}}\right) 100\%$ where S_{best} is the known good solution for the instance and S is the score of the test run. In one instance the solution was better than both the reference good solution and the result of the experiment done by Souffriau et al. (2013). For this instance the gap value is printed in bold.

Test runs were performed with 23 parallel workers. The workload was partitioned by assigning a fixed number of 2000 iterations to each worker.

Table A.13: Gap results for cooperative GRASP-ILS hybrid: MCTOPMTW ($m = 1$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	320.00	0.00%	0.00%	0.00%
c102	1236	360.00	0.00%	0.00%	0.00%
c103	1236	400.00	0.00%	0.00%	0.00%
c104	1236	420.00	0.00%	0.00%	0.00%
c105	1236	340.00	5.88%	5.88%	5.88%
c106	1236	340.00	0.00%	0.00%	0.00%
c107	1236	370.00	0.00%	0.00%	0.00%
c108	1236	370.00	0.00%	0.00%	0.00%
c109	1236	380.00	0.00%	0.00%	0.00%
r101	230	198.00	3.03%	3.03%	3.03%
r102	230	286.00	0.00%	0.00%	0.00%
r103	230	293.00	0.00%	0.00%	0.00%

[4] `http://www.mech.kuleuven.be/en/cib/op/`

Table A.13: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
r104	230	303.00	5.28%	5.28%	5.28%
r105	230	247.00	0.00%	0.00%	0.00%
r106	230	293.00	0.00%	0.00%	0.00%
r107	230	299.00	4.35%	4.35%	4.35%
r108	230	308.00	7.14%	7.14%	7.14%
r109	230	277.00	6.14%	6.14%	6.14%
r110	230	284.00	5.99%	6.48%	10.92%
r111	230	297.00	5.05%	5.05%	5.05%
r112	230	298.00	0.00%	0.00%	0.00%
rc101	240	219.00	1.37%	1.37%	1.37%
rc102	240	266.00	0.00%	0.00%	0.00%
rc103	240	266.00	0.00%	0.00%	0.00%
rc104	240	301.00	6.64%	6.64%	6.64%
rc105	240	244.00	0.00%	0.00%	0.00%
rc106	240	252.00	0.00%	0.00%	0.00%
rc107	240	277.00	0.00%	0.00%	0.00%
rc108	240	298.00	0.00%	0.00%	0.00%
pr01	1000	308.00	0.00%	0.00%	0.00%
pr02	1000	404.00	0.00%	0.10%	0.99%
pr03	1000	394.00	0.00%	0.84%	4.82%
pr04	1000	489.00	0.00%	1.96%	13.29%
pr05	1000	595.00	0.00%	0.00%	0.00%
pr07	1000	298.00	6.71%	6.71%	6.71%
pr08	1000	463.00	4.54%	5.29%	8.21%
pr09	1000	493.00	0.00%	0.61%	6.09%

Table A.14: Gap results for cooperative GRASP-ILS hybrid: MCTOPMTW ($m = 2$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	590.00	3.39%	3.39%	3.39%
c102	1236	650.00	3.08%	3.08%	3.08%
c103	1236	700.00	1.43%	3.57%	4.29%
c104	1236	750.00	0.00%	2.00%	4.00%
c105	1236	640.00	3.12%	4.53%	4.69%
c106	1236	620.00	1.61%	1.61%	1.61%

Table A.14: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c107	1236	670.00	4.48%	5.22%	5.97%
c108	1236	670.00	0.00%	2.39%	2.99%
c109	1236	710.00	2.82%	4.93%	5.63%
r101	230	330.00	0.00%	0.00%	0.00%
r102	230	508.00	0.00%	0.22%	2.17%
r103	230	513.00	0.00%	1.01%	3.51%
r104	230	539.00	0.00%	0.98%	5.01%
r105	230	430.00	2.56%	2.56%	2.56%
r106	230	529.00	5.67%	7.49%	11.34%
r107	230	529.00	0.00%	4.93%	7.94%
r108	230	549.00	4.01%	5.72%	7.65%
r109	230	498.00	8.03%	10.06%	12.05%
r110	230	515.00	7.38%	8.29%	9.32%
r111	230	535.00	0.00%	3.64%	8.22%
r112	230	515.00	0.00%	2.78%	8.93%
rc101	240	427.00	0.70%	0.94%	3.04%
rc102	240	494.00	-0.20%	5.49%	8.10%
rc103	240	519.00	4.43%	6.01%	7.71%
rc104	240	565.00	2.48%	4.48%	6.90%
rc105	240	459.00	0.00%	2.51%	7.84%
rc106	240	458.00	0.66%	0.66%	0.66%
rc107	240	515.00	0.00%	0.00%	0.00%
rc108	240	546.00	0.00%	2.01%	7.33%
pr01	1000	471.00	2.12%	5.27%	7.43%
pr02	1000	660.00	4.24%	5.97%	7.88%
pr03	1000	714.00	3.22%	6.50%	14.57%
pr04	1000	863.00	4.87%	7.75%	12.40%
pr05	1000	1011.00	6.33%	8.89%	13.06%
pr07	1000	552.00	3.08%	3.51%	3.62%
pr08	1000	796.00	4.77%	8.23%	10.80%
pr09	1000	867.00	11.07%	15.51%	18.69%

Table A.15: Gap results for cooperative GRASP-ILS hybrid: MCTOPMTW ($m = 3$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	790.00	1.27%	1.27%	1.27%
c102	1236	890.00	1.12%	1.12%	1.12%
c103	1236	960.00	2.08%	3.02%	3.12%
c104	1236	1010.00	2.97%	4.36%	4.95%
c105	1236	840.00	3.57%	4.05%	4.76%
c106	1236	840.00	2.38%	3.10%	3.57%
c107	1236	900.00	5.56%	8.11%	10.00%
c108	1236	900.00	5.56%	6.78%	7.78%
c109	1236	950.00	5.26%	6.32%	7.37%
r101	230	481.00	2.70%	3.26%	5.41%
r102	230	685.00	2.92%	4.09%	5.99%
r103	230	720.00	5.56%	7.06%	8.47%
r104	230	765.00	2.48%	3.92%	7.45%
r105	230	609.00	3.78%	4.60%	6.73%
r106	230	719.00	6.54%	12.24%	15.16%
r107	230	747.00	2.41%	4.27%	6.16%
r108	230	790.00	2.53%	8.63%	11.39%
r109	230	699.00	3.58%	8.83%	12.45%
r110	230	711.00	5.06%	9.04%	13.92%
r111	230	764.00	6.94%	9.21%	12.17%
r112	230	758.00	3.03%	6.36%	10.55%
rc101	240	604.00	7.45%	9.30%	10.60%
rc102	240	698.00	**-1.29%**	6.76%	10.74%
rc103	240	747.00	2.54%	6.97%	12.05%
rc104	240	822.00	5.23%	7.29%	10.95%
rc105	240	654.00	3.36%	7.13%	10.55%
rc106	240	678.00	2.06%	5.90%	8.11%
rc107	240	745.00	2.55%	4.19%	5.77%
rc108	240	757.00	1.32%	5.27%	8.45%
pr01	1000	598.00	5.85%	8.16%	10.20%
pr02	1000	899.00	5.23%	7.17%	8.23%
pr03	1000	946.00	6.45%	8.60%	12.05%
pr04	1000	1195.00	7.28%	9.44%	11.80%
pr05	1000	1356.00	2.73%	4.67%	7.01%
pr07	1000	713.00	3.79%	6.96%	8.70%
pr08	1000	1082.00	6.10%	7.82%	10.26%

Table A.15: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr09	1000	1144.00	4.55%	8.28%	11.98%

Table A.16: Gap results for cooperative GRASP-ILS hybrid: MCTOPMTW ($m = 4$)

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
c101	1236	1000.00	2.00%	2.70%	3.00%
c102	1236	1090.00	0.92%	0.92%	0.92%
c103	1236	1150.00	2.61%	4.09%	5.22%
c104	1236	1220.00	3.28%	4.26%	4.92%
c105	1236	1030.00	3.88%	4.27%	4.85%
c106	1236	1040.00	4.81%	5.29%	5.77%
c107	1236	1100.00	5.45%	6.09%	6.36%
c108	1236	1100.00	5.45%	6.09%	6.36%
c109	1236	1180.00	5.08%	7.29%	8.47%
r101	230	601.00	1.66%	2.16%	3.16%
r102	230	807.00	2.97%	5.13%	7.31%
r103	230	878.00	1.59%	7.08%	9.00%
r104	230	941.00	2.55%	5.56%	8.08%
r105	230	735.00	2.45%	5.61%	9.12%
r106	230	870.00	6.55%	8.90%	10.80%
r107	230	927.00	6.80%	10.22%	12.84%
r108	230	982.00	4.89%	8.03%	10.69%
r109	230	866.00	9.12%	11.71%	14.20%
r110	230	870.00	7.24%	9.55%	11.26%
r111	230	935.00	6.20%	9.20%	11.76%
r112	230	939.00	4.47%	7.85%	10.65%
rc101	240	794.00	5.79%	10.20%	13.60%
rc102	240	881.00	10.78%	12.76%	13.85%
rc103	240	947.00	6.02%	9.43%	13.31%
rc104	240	1019.00	3.63%	5.29%	6.87%
rc105	240	841.00	2.26%	9.77%	12.96%
rc106	240	874.00	5.49%	8.09%	9.84%
rc107	240	951.00	5.05%	8.91%	11.04%
rc108	240	998.00	4.41%	7.66%	11.42%
pr01	1000	644.00	1.55%	2.11%	2.95%

Table A.16: Continued

Problem	d_{lim}	*Best known*	*Min.*	*Avg.*	*Max.*
pr02	1000	1014.00	3.85%	4.29%	4.93%
pr03	1000	1162.00	4.39%	8.90%	11.36%
pr04	1000	1452.00	5.72%	7.29%	8.82%
pr05	1000	1665.00	4.86%	7.43%	9.61%
pr07	1000	840.00	6.31%	7.87%	9.05%
pr08	1000	1267.00	7.42%	9.06%	10.02%
pr09	1000	1460.00	9.32%	10.70%	12.60%

A.5 Time Dependent Orienteering Problem

Verbeeck et al. (2014) designed a dataset for the TDOP. For evaluation, we use problems 1-3 from the dataset that have been solved to optimality using CPLEX. The results of the cooperative GRASP-ILS hybrid (`GRILS-T`) are given in Table A.17.

t_{max} is the time budget for the given instance. *Optimum* is the result of the CPLEX solver, using the time dependent arc traversal speeds from the dataset. *Min.*, *Avg.* and *Max.* give the minimum, arithmetic mean and maximum gap for the given instance over 10 test runs. The gap was calculated as $\left(1 - \frac{S}{S_{opt}}\right) 100\%$ where S_{opt} is the optimum solution for the instance and S is the score of the test run.

Test runs were performed with 23 parallel workers. The workload was partitioned by assigning a fixed number of 10000 iterations to each worker.

Table A.17: Gap results for cooperative GRASP-ILS hybrid: Time Dependent Orienteering Problem

Instance	t_{max}	*Optimum*	*Min.*	*Avg.*	*Max.*
1.a	5	115.00	0.00%	6.09%	8.70%
1.b	6	135.00	3.70%	5.93%	7.41%
1.c	7	160.00	6.25%	6.56%	9.38%
1.d	8	185.00	5.41%	5.68%	8.11%
1.e	9	210.00	7.14%	8.10%	9.52%

Table A.17: Continued

Instance	t_{max}	*Optimum*	*Min.*	*Avg.*	*Max.*
1.f	10	230.00	4.35%	7.39%	8.70%
1.g	11	250.00	4.00%	7.20%	8.00%
1.h	12	270.00	5.56%	8.89%	11.11%
2.a	5	100.00	0.00%	0.00%	0.00%
2.b	6	150.00	10.00%	10.00%	10.00%
2.c	7	195.00	15.38%	16.41%	17.95%
2.d	8	220.00	0.00%	8.64%	11.36%
2.e	9	260.00	7.69%	10.38%	11.54%
2.f	10	310.00	6.45%	8.55%	12.90%
2.g	11	340.00	0.00%	5.59%	8.82%
2.h	12	375.00	1.33%	5.60%	8.00%
2.i	13	425.00	2.35%	6.12%	9.41%
3.a	5.5	370.00	5.41%	8.92%	10.81%
3.b	6.5	420.00	2.38%	4.29%	4.76%
3.c	7.5	500.00	4.00%	10.20%	14.00%
3.d	8.5	560.00	10.71%	13.93%	16.07%
3.e	9.5	620.00	12.90%	15.81%	17.74%
3.f	10.5	650.00	12.31%	15.69%	16.92%
3.g	11.5	690.00	11.59%	14.78%	17.39%