Università
Ca'Foscari
Venezia

**Department
of Management**

Working Paper Series

P. Pellegrini, L. Castelli, and R. Pesenti

Metaheuristic algorithms for the simultaneous slot allocation problem

Working Paper n. 9/2011
October 2011

ISSN: 2239-2734

Metaheuristic Algorithms for the Simultaneous Slot Allocation Problem

Paola Pellegrini[1], Lorenzo Castelli[2] and Raffaele Pesenti[3]

[1] IRIDIA-CoDE, Université Libre de Bruxelles, Belgium
[2] Università di Trieste, Italy
[3] Università Ca' Foscari Venezia, Italy

Abstract

In this paper, we formalize the simultaneous slot allocation problem. It is an extension of the problem currently tackled for allocating airport slots: it deals with all airports simultaneously and it enforces the respect of airspace sector capacities. By solving this novel problem, the system may overcome some major inefficiencies that characterize the current slot allocation process. We tackle the simultaneous slot allocation problem with two algorithms based on metaheuristics, namely Iterated Local Search and Variable Neighborhood Search, and with an integer linear programming model: for each of these three algorithms, we allow a fixed computation time, and we take the best solution found during that time as the final solution. We compare these algorithms on randomly generated instances, and we show that, when small instances are to be tackled, metaheuristics are competitive with the exact model. When medium or large instances are to be tackled, the exact model suffers some major issues in terms of memory and computation time requirements. Metaheuristics, instead, can deal with very large instances, achieving very high quality results.

Air Traffic Management; Airport slot allocation; Metaheuristics; Integer linear programming

1 Introduction

Forecasts predict that in the next decades the capacity offered by two major pieces of the aviation infrastructure, the airports system and the air traffic management (ATM) system (Barnhart *et al.* , 2003), will increase at a lower rate than the expected worldwide steady growth of air traffic demand, unless major actions are taken by air transport authorities and stakeholders (Eurocontrol Experimental Centre, 2008). The modernization of the ATM infrastructure to meet the requirements of airspace users while guaranteeing the adequate level of safety and contributing to the sustainable development of the air transport

system, is the main target of the SESAR and NextGen initiatives launched in Europe and the United States, respectively (see, e.g., Brooker, 2008). For instance, "the Single European Sky performance objectives aim at tripling capacity, reducing ATM costs by half, improving safety by a factor of 10 and reducing the environmental impact of each flight by 10%" (European Commission, 2010).

The building of new capacity is an objective harder to achieve in the airport context because physical, environmental and political constraints limit the construction of new terminals and runways. Airport capacity is also managed through administrative measures which unfortunately appear to be quite inefficient even for the today's level of traffic (Airport Council International Europe, 2004). In particular, airport capacity can be expressed in terms of slots: a *slot* is the permission given to a carrier to use the full range of airport infrastructure necessary to operate on a specific date and time for landing or taking-off (European Commission, 1993). The inefficiency of the current *slot allocation process* manifests itself in terms of: unsatisfied or unaccommodated demand, late return of unwanted slots, flights operated off slot times ("off slot"), and failure to operate allocated slots ("no shows"). These issues are, at least in part, due to the infeasibility of the predefined schedule and to the fact that this infeasibility is detected shortly before the day of operations. According to the estimations presented in the report by Airport Council International Europe (2009), large congested European airports lose a revenue of 20 million Euros per season due to late slot returns that do not allow for substitution or redistribution. The impact of this inefficient airport capacity management will be more and more evident in the future as, by 2030, the demand is expected to exceed airport capacities of about 2.3 million flights (Eurocontrol Experimental Centre, 2008). To limit this impact, the slot allocation process must undergo some major revisions.

The slot allocation process follows the rules and principles described in Regulation 95/1993 of the European Commission (1993) and its subsequent amendments (European Commission, 2002, 2003, 2004, 2007, 2008, 2009). They are the evolution of a system created by the International Air Transport Association (IATA) in 1947. In this process, the European Union Member States identify the most congested airports and denote them as *coordinated*. These are the airports where an airport coordinator must allocate a slot to an airline, before the airline itself is actually allowed exploiting the corresponding facilities (European Commission, 1993). Each coordinated airport has its own coordinator that fulfills multiple tasks (European Commission, 1993; IATA, 2011). She:

1. establishes the declared airport capacity by fixing the number of slots available per time unit;

2. guarantees the grandfather rights, i.e., the rights of airlines to exploit the slots they have actually used in the preceding equivalent season. In particular, she implements the use-it-or-lose-it rule by, on one side, identifying as slots subject to grandfather rights the ones used for no less than 80% of the time during the previous season; on the other side, considering free from grandfather rights all the other slots;

3. allocates series of slots to airlines according to the following steps: first she allocates the grandfathered slots (slots on which grandfather rights exist), second she allocates half of the non-grandfathered slots to new entrants (airlines with limited presence at a coordinated airport), third she allocates the remaining slots to unallocated requests according to their subordinate priority (IATA, 2011);

4. warrants the operational and legal feasibility of subsequent slot exchanges between airlines.

Task 3 constitutes the so-called *primary allocation*. This phase of the slot allocation process is initially pursued by each airport independently from the others, but then it is discussed and adjusted by the representatives of airports and airlines meeting at an IATA conference. After this conference, bilateral negotiations leading to slot exchanges, more commonly referred to as *secondary trading*, may continue among airlines (European Commission, 2008).

Several studies focusing on the modernization of the slot allocation process appear in the literature. Some of them (DotEcon Ltd, 2001; Kösters, 2007; NERA Economic Consulting, 2004) underline the necessity of allocating coherently the slots at origin and destination airports of each flight. In fact, in the adjustment that follows the primary allocation some inefficiencies may arise and these are not always corrected in the secondary trading. For example, the airport coordinators may not be able to allocate slots coherently, that is, so that the slots that can be coupled and exploited by a flight following an existing route. Thus, it may occur that some flights from/to coordinated airports receive no slots even if some slots will remain unused. Still, the great majority of the studies either on the primary allocation (DotEcon Ltd, 2001, 2006; Fukui, 2010; Kleit & Kobayashi, 1996; Maldoom, 2003; Sentance, 2003; Starkie, 1998; Verhoef, 2010) or on the secondary trading (DotEcon Ltd, 2001; Mott MacDonald Limited, 2006; de Wit & Burghouwt, 2007; UK Civil Aviation Authority, 2001; Holt *et al.* , 2007; NERA Economic Consulting, 2004) do not deal with this necessity. Most of these works stress the need of an improved slot allocation process for maximally exploit capacity, enhancing competition and reducing the existing barriers to entry. To this aim they try to quantify the inefficiency and the optimization potentials of the current process. The remaining papers focus on the qualitative investigation of the opportunity of implementing auction mechanisms following the current practice at some US airports. The first authors considering the interdependence among airports are Rassenti *et al.* (1982). They propose a combinatorial auction of the slots. Yet, the main focus of their paper is on the efficiency and robustness of the auction design in terms of demand revelation. More recently, Castelli *et al.* (2010) and Pellegrini *et al.* (2011) have considered the interdependency of slots at different airports respectively for the primary allocation and the secondary trading. To tackle the primary allocation Castelli *et al.* (2010) adapted a linear integer programming model that was previously presented as a tool for solving the air traffic flow management problem in the day of operations (Bertsimas *et al.* , 2011). Pellegrini *et al.* (2011) extended this model to use it in a combinatorial exchange

mechanism for the secondary trading.

In this paper, we focus on the accomplishment of a simultaneous coherent primary allocation at the different airports which also takes into account the route flown by aircrafts to the enforcement of the respect of airspace *sectors* capacity. Despite is it well-known that one of the limitations to the increase of many European airports' capacity is given by the congestion in airspace (SESAR Consortium, 2008), sector capacities have been considered so far just in Pellegrini *et al.* (2011). We call this problem simultaneous slot allocation problem (SSAP). The SSAP differs from the one currently tackled by coordinators in their task 3 as:

- we solve the primary allocation problem at several airports simultaneously;
- we seek a slot allocation that meets sector capacity constraints.

The first target of the SSAP is to find an allocation that will not reveal itself to be infeasible on the day of operations, in absence of unexpected disruption of the system such as bad weather conditions limiting sector capacities. Then among all feasible allocations the SSAP considers two hierarchically ordered objectives: first it maximizes the number of flights to which slots are allocated (*accommodated flights*); second it minimizes the cost of receiving slots different from the originally requested ones (*ideal slots*).

We introduce and compare three methods for dealing with the SSAP. Here note, in this paper we do not consider airlines priorities in terms of either grandfather rights or new entrants privileges: we leave this as a further research step.

The methods we propose for tackling the SSAP are an integer linear programming model and two metaheuristic algorithms. We refer to the former as the Truncated Integer Linear Programming model (TILP): it is a further extension of the model presented in Castelli *et al.* (2010) which takes into account sector capacities, and allows the existence of solutions that do not accommodate all requested flights. It is truncated in the sense that only a predetermined amount of time is allowed for the computation. The metaheuristic algorithms are based on Iterated Local Search (ILS) (Lourenço *et al.* , 2003) and Variable Neighborhood Search (VNS) (Hansen & Mladenovič, 2001), and they are specifically designed for tackling the SSAP.

In the experimental analysis we test the three methods on randomly generated instances. We compare the results of these methods with the ones achieved by two benchmark algorithms. The first one mimics the current primary allocation procedure. This benchmark allows us to verify whether our heuristic solution to the SSAP is preferable to the current primary allocation process procedure, in terms of number of flights that can be performed. The second one is a trivial random restart local search algorithm. It shares the local procedure search with ILS and VNS, but it does not include any further expedient for smartly exploring the search space. This benchmark allows us to understand whether the results achieved are due solely to the local search procedures, or rather to the metaheuristics as a whole.

Our analysis shows that TILP is a very good option as far as the instances to be tackled are rather small and the computational time is not an issue. When, instead, one tackles large instances or needs to find a solution in quite a short time, TILP is not a viable option. In this case, metaheuristics are a valuable alternative: they can deal with very large instances, and they return good quality solutions starting from the very first seconds of computation. Metaheuristics were able to accommodate all flights and to find either an optimal or a very good sub-optimal solution in all the instances in which TILP was able to solve SSAP exactly. Metaheuristics were still able to obtain solutions accommodating almost all flights, when the instances were too large to be tackled with TILP.

The rest of the paper is organized as follows. In Section 2 we formally state the characteristics of the SSAP. In Section 3 and 4 we describe the integer linear programming model and metaheuristic algorithms, respectively. In Section 5 we present the algorithms that we use as benchmarks in the experimental analysis. In Section 6 we present the experimental setup used in the analysis, and in Section 7 we report the results achieved. Finally, in Section 8 we draw some conclusions and we delineate future research developments.

2 The simultaneous slot allocation problem

The SSAP consists in allocating slots to airlines so that the maximum number of flights are accommodated, and, if multiple equivalent solutions with the maximum number of flights accommodated exist, the minimum *shift cost* is imposed to airlines. The shift cost occurs when a flight is assigned a slot different from the ideal one. This cost may manifest itself in terms of either revenue losses from unsold passenger tickets or additional organizational costs, due for example to the impossibility of implementing a minimum cost crew scheduling.

The slot allocation must meet multiple types of requirements:

- **capacity requirements**: neither airport nor sector capacity must ever be exceeded;
- **route requirements**: slots must be allocated so that, for each accommodated flight, a route of the appropriate duration exists for connecting the origin and destination airports at the allocated time;
- **time requirements**: slots allocated to a flight, if any, must be within an acceptable time period indicated by the airline: for example, a slot cannot be allocated to an airline if it is more than 30 minutes later than the airline's ideal slot;
- **duration requirements**: route duration cannot be longer than a predefined value: for example, slots cannot be allocated to an airline if they imply that its flight must be rerouted, adding 3 hours to its shortest route.

We express all routes in terms of sequence of slots, by extending the concept of slot to both airports that are not coordinated, and to sectors. This extension allows us to control the respect of sector capacity and route requirements.

3 Truncated integer linear programming model

In this section we introduce the Truncated integer linear programming algorithm for the SSAP. TILP models the SSAP as an integer linear programming problem. It tries to solve exactly this integer linear programming problem within a predetermined computation time. If no optimal solution is found within such a time, TILP returns the best feasible solution found so far, if any.

Within TILP, and even in the other algorithms that we present in this paper, we subdivide the time horizon in *time intervals* of fixed length: all slots have the duration of one time interval, and all sectors are sized so that flights can traverse them in one time interval. Then, from a mathematical perspective, each slot is a pair (j,t) with $j \in K \cup S$ and $t \in T$, being K the set of airports, S the set of sectors, and T the set of time intervals. A route r is a set of slots (j,t), $j \in K \cup S$ and $t \in T$, connecting the origin to the destination of a flight.

The further notation that we need to introduce is the following:

$A \equiv$ set of airlines,

$F \equiv$ set of flights,

$F_a \subseteq F \equiv$ set of flights of airline $a \in A$,

$K_{j,t} \equiv$ capacity of airport $j \in K$ at time interval $t \in T$,

$S_{j,t} \equiv$ capacity of sector $j \in S$ at time interval $t \in T$,

$S^f \subseteq S \cup K \equiv$ set of sectors that can be flown by flight f, including the origin and the destination airports,

$P_i^f \equiv$ set of sector i's preceding sectors for flight f ($i \in S^f$),

$L_i^f \equiv$ set of sector i's subsequent sectors for flight f ($i \in S^f$),

$l_{f,j,j'} \equiv$ number of time intervals that flight f must spend in sector j before entering in sector j',

$end_f \equiv$ maximum acceptable duration of flight f.

$dt_f \equiv$ ideal departure time interval of flight $f \in F$,

$at_f \equiv$ ideal arrival time interval of flight $f \in F$,

$orig_f \equiv$ origin airport of flight $f \in F$,

$dest_f \equiv$ destination airport of flight $f \in F$,

$T^f_{orig_f} = [\underline{T}^f_{orig_f}, \bar{T}^f_{orig_f}] \equiv$ set of time intervals declared acceptable for the departure flight $f \in F_a$ by airline a: $\underline{T}^f_{orig_f} \equiv$ earliest departure time declared acceptable, $\bar{T}^f_{orig_f} \equiv$ latest departure time declared acceptable,

$T^f_{dest_f} = [\underline{T}^f_{dest_f}, \bar{T}^f_{dest_f}] \equiv$ set of arrival time interval declared acceptable for the arrival of for flight $f \in F_a$ by airline a: $\underline{T}^f_{dest_f} \equiv$ earliest arrival time declared acceptable, $\bar{T}^f_{dest_f} \equiv$ latest arrival time declared acceptable,

c_t^f ≡ cost of having flight $f \in F$ arriving at time interval t,

c_{dur}^f ≡ cost of increasing the duration for flight $f \in F$ of one time interval.

M ≡ large constant, referred as to"big$M$".

Given the above notation, in the following we introduce the decision variables, objective function and constraints that define the integer linear programming model for the SSAP. As already pointed out in the Introduction, this model is similar to the ones presented in Castelli *et al.* (2010) and Pellegrini *et al.* (2011).

Decision variables

For all flights $f \in F$ and all slots $(j,t) \in (K \cup S) \times T$, we consider the following binary decisional variables:

$$w_{j,t}^f = \begin{cases} 1 & \text{if a slot } (j,t') \text{ with } t' \leq t, \text{ is allocated to flight } f, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In addition, for all flights $f \in F$ and all sector slots $(j,t) \in S \times T$, we also introduce the binary variables:

$$co_{j,t}^f = \begin{cases} 1 & \text{if flight } f \text{ is allocated to slot } (j,t), \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Objective function

The objective function is the weighted sum of two components: the number of flights accommodated and the overall shift cost imposed to the flights. Given the definition of decision variables $w_{j,t}^f$'s, if a flight f is accommodated then $w_{dest_f, \bar{T}_{dest_f}^f}^f$ equals 1. Thus, the first component of the objective function can be formulated as

$$\sum_{f \in F} w_{dest_f, \bar{T}_{dest_f}^f}^f .$$

For what concerns the second component, that is the total shift cost, let C_f be the shift cost imposed to flight f:

Figure 1: Example of possible slot allocations implying different costs.

$$C_f = \sum_{t \in T^f_{dest_f}} c^f_t (w^f_{dest_f,t} - w^f_{dest_f,t-1}) + c^f_{dur} \left[\sum_{t \in T^f_{dest_f}} t(w^f_{dest_f,t} - \right. \tag{3}$$

$$\left. - w^f_{dest_f,t-1}) - \sum_{t \in T^f_{orig_f}} t(w^f_{orig_f,t} - w^f_{orig_f,t-1}) - (at_f - dt_f) \right].$$

The first part of cost (3) penalizes an arrival time interval t different from the desired one at_f. The second part of (3) penalizes the flight duration, if longer than the desired one $at_f - dt_f$. Finally, the combination of the two terms implicitly penalizes also a departure time interval different from the desired one. The cost C_f is null when the flight f is assigned to its ideal arrival slot and its flight duration is the planned one.

Following Castelli *et al.* (2010) and Pellegrini *et al.* (2011), for all $f \in F$, we assume that the cost associated to the increase of flight duration is proportional to this duration, i.e., c^f_{dur} is constant. Instead, we assume that the cost function associated to the arrival shift increases more than proportionally with respect to the duration of the shift: in particular, it has structure $c^f_t = \alpha_f |t - at_f|^{\beta_f}$ where α_f and β_f are nonnegative parameters, with $\alpha_f < c^f_{dur}$ and $\beta_f > 1$. By setting $\alpha_f < c^f_{dur}$ we ensure that an unitary arrival shift is preferred to an unitary increase of flight duration. See Figure 1 for an example of the computation of C_f. The condition $\beta_f > 1$ ensures that shifting the arrival of one flight of two time intervals has a higher cost than shifting the arrival of two flights of one time interval each.

The overall objective function is the following:

$$z^* = \max \sum_{f \in F} \left(M w^f_{dest_f, \bar{T}^f_{dest_f}} - C_f \right). \tag{4}$$

The presence of the big$M$ coefficient in the objective functions is a possible way to impose the hierarchy present between the two objectives of the SSAP.

The Constraints

The integer linear programming model includes the following constraints:

$$w^f_{orig_f, \underline{T}^f_{orig_f}-1} = 0 \qquad \forall f \in F \tag{5}$$

$$w^f_{dest_f, \bar{T}^f_{dest_f}} - w^f_{dest_f, \bar{T}^f_{dest_f}+1} = 0 \qquad \forall f \in F \tag{6}$$

$$\sum_{f \in F: dest_f = j \vee orig_f = j} w^f_{j,t} - w^f_{j,t-1} \leq K_{j,t} \qquad \forall j \in K,\, t \in T \tag{7}$$

$$co^f_{j,t} \geq w^f_{j,t} - \sum_{i \in L^f_j : t \leq \bar{T}^f_i} w^f_{i,t} \qquad \forall f \in F,\, j \in S^f \setminus \{dest_f\},\, t \in T \tag{8}$$

$$\sum_{f \in F: j \in S^f} co^f_{j,t} \leq S_{j,t} \qquad \forall j \in S,\, t \in T \tag{9}$$

$$w^f_{j,t} \leq \sum_{i \in L^f_j} w^f_{i,t+l_{f,j,i}} \qquad \forall f \in F,\, j \in S^f \setminus \{dest_f\},\, t \in T \tag{10}$$

$$w^f_{j,t} \leq \sum_{i \in P^f_j} w^f_{i,t-l_{f,i,j}} \qquad \forall f \in F,\, j \in S^f \setminus \{orig_f\},\, t \in T \tag{11}$$

$$w^f_{j,\bar{T}^f_j} \leq \sum_{i \in L^f_j} w^f_{i,\bar{T}^f_i} \qquad \forall f \in F,\, j \in S^f \setminus \{dest_f\}, \tag{12}$$

$$\sum_{i \in L^f_j} w^f_{i,\bar{T}^f_i} \leq 1 \qquad \forall f \in F,\, j \in S^f \setminus \{dest_f\}, \tag{13}$$

$$w^f_{orig_f,t} - w^f_{dest_f,t+end_f} \leq 0 \qquad \forall f \in F,\, t \in T \tag{14}$$

$$w^f_{j,t-1} - w^f_{j,t} \leq 0 \qquad \forall f \in F,\, j \in S^f,\, t \in T \tag{15}$$

Constraints (5) and (6) ensure the respect of time requirements, that is, that no flight is accommodated outside the time interval that is declared to be acceptable by the airline. Constraints (7), (8) and (9) impose the respect capacity requirements, for what concerns airports – Constraints (7) – and sectors – Constraints (8) and (9). Constraints (10) and (11), and (12), (13) and (15) guarantee the routes' time and spatial coherence, respectively. All together, these constraints impose the respect of route requirements. As an example, constraints (11) ensure that a route assigned to an aircraft cannot make it enter a sector without having crossed a preceding one. Finally, Constraints (14) warrant the respect of duration requirements.

As it is done in Bertsimas *et al.* (2011), Castelli *et al.* (2010) and Pellegrini *et al.* (2011), we also implemented valid inequalities for speeding up the solution process.

4 Metaheuristic algorithms

In this section, we introduce two algorithms to address the SSAP, which are respectively based on Iterated Local Search (Lourenço *et al.* , 2003) and on Variable Neighborhood Search (Hansen & Mladenovic̀, 2001) metaheuristics. Both algorithms receive as input the structure of the network and the route and costs characterizing each flight. They return, again for each flight, a boolean value indicating whether the flight is accommodated or not and, in case of positive answer, all the slots it uses (in terms of combination of airport/sector and time).

Both ILS and VNS rely on the same two local search procedures. The first local search (flight-local-search) aims at increasing the number of flights accommodated. The second local search (cost-local-search) aims at decreasing the solution cost, given the set of flights accommodated. The pseudocodes of these two procedures are respectively presented in Figures 2 and 3. Both procedures are randomized[1] first-improvement local searches (Hoos & Stützle, 2004), and they reiterate recursively as far as they find an improved solution. The performance of these search procedures depend on two tuning parameters: the number of unsuccessful trials t to be completed before considering the current solution a local minimum; the number of flights q to be either de-accommodated (in the flight-local-search) or randomly shifted at the beginning of each trial (in the cost-local-search).

Both ILS and VNS must periodically generate a random solution. To this aim, flights are randomly ordered and then, starting from the first one, they are accommodated one by one as far as their ideal slots are available and there is enough capacity left in the sectors belonging to one of their minimum-duration routes. The pseudocode of this procedure is illustrated in Figure 4.

The ILS algorithm starts from a first randomly generated feasible solution and it explores the solution space by iteratively calling the two local search procedures. The algorithm makes multiple local search calls before considering the search of the current region completed. After multiple calls, it perturbs the best solution found so far for identifying the new solution for starting local search. The perturbation allows escaping the basin of attraction of a current local minimum and, at the same time, ensures that the search remains focused on the best region identified so far. When the local search procedures are not able to find a better solution, the algorithm restarts from a new randomly generated solution. The ratio at the basis of this metaheuristic is the so called massif central phenomenon (Fonlupt *et al.* , 1999) that assumes that the best local optima are near to the global optimum. The magnitude according to which this phenomenon emerges in a specific problem is very hard to quantify. Still, ILS appears very well performing in a wide set of different problems (Hoos & Stützle, 2004). The pseudocode of ILS is introduced in Figure 5. ILS must be tuned with respect to three main parameters: the number r of iterations without an improvement, in terms of number of flights accommodated, that can be

[1]From here on, when we use the notion of random selection we mean that the sample is performed based on a uniform probability distribution.

```
S=initial solution
trial=0
while (trial < t & time left) do
  S'=S after de-accommodating q randomly drawn flights
  for (f in the set of flights non-accommodated in S', considered in random order) do
    if (capacity in f's ideal slots at airports is available) then
      if (capacity along a route of f's is available) then
        accommodate f along that route
        next f
    else
      Comp^f_orig = set of flights competing with f in orig_f
      Comp^f_dest = set of flights competing with f in dest_f
      if (Comp^f_orig not empty) then
        for (f' in Comp^f_orig, considered in random order) do
          if (Comp^f_dest not empty) then
            for (f'' in Comp^f_dest, considered in random order) do
              randomly shift f' and f''
              if (capacity in f's ideal slots at airports is available) then
                if (capacity along a route of f's is available) then
                  accommodate f along that route
                  next f
          else
            for (f' in Comp^f_orig, considered in random order) do
              randomly shift f'
              if (capacity in f's ideal slots at airports is available) then
                if (capacity along a route of f's is available) then
                  accommodate f along that route
                  next f
      else if (Comp^f_dest not empty) then
        for (f'' in Comp^f_dest, considered in random order) do
          randomly shift f''
          if (capacity in f's ideal slots at airports is available) then
            if (capacity along a route of f's is available) then
              accommodate f along that route
              next f
      else
        if (capacity along a route of f's is available) then
          accommodate f along that route
          next f
  if (number of flights in S' > number of flights in S) then
    S=S'
    if (all flights are accommodated in S) flight-local-search(S)
    else cost-local-search(S)
  trial=trial +1
return S
```

Figure 2: Pseudocode of the flight-local-search procedure.

```
S=initial solution
trial=0
while (trial < t & time left) do
  S'=S after randomly shifting q randomly drawn flights
  for (f in the set of flights accommodated in S', considered in random order) do
    if (C_f > 0) then
      Comp^f = set of flights f' such that C_f' < C_f, competing with f in orig_f or dest_f
      for (f' in the Comp^f, considered in random order) do
        randomly shift f'
        if (f can be shifted) then
          shift f so that its cost is decreased
          set C_previous = C_f' + C_f
          compute C_f' and C_f
          if (C_f' + C_f > C_previous)
            set f and f' to their initial slots
          else
            next f
        else
          set f' to its initial slots
  shorten all flights for which it is possible
  if (cost of flights in S' > number of flights in S) then
    S=S'
    cost-local-search(S)
  trial=trial +1
return S
```

Figure 3: Pseudocode of the cost-local-search procedure.

```
S = ∅
for (f in the set of all flights, considered in random order) do
  if (capacity in f's ideal slots at airports is available &
    & capacity along a route of f's is available) then
      accommodate f along that route ⇒ S = S ∪ {(f, ideal_lots_f, route_f)}
next f
return S
```

Figure 4: Pseudocode of the procedure used for generating a random solution.

```
S* = S = randomly drawn solution
while (time left) do
  if (no improvement in the last r iterations & not all flights are accommodated) then
    S=randomly drawn solution
  if (not all flights are accommodated)
    for (round in 1:k) S = flight-local-search(S)
    S=S after de-accommodating p randomly drawn flights
  else
    for (round in 1:k) S = cost-local-search(S)
    S=S after randomly shifting p randomly drawn flights
  if S is better than S* then S* = S
return S*
```

Figure 5: Pseudocode of the ILS algorithm.

```
S* = S = randomly drawn solution
q0=q
while (time left) do
  if (no improvement in the last r iterations & not all flights are accommodated) then
    S=randomly drawn solution
    q = q0
  q = q + i
  if (not all flights are accommodated)
    for (round in 1:k) S = flight-local-search(S)
  else
    for (round in 1:k) S = cost-local-search(S)
  if S is better than S* then S* = S
return S*
```

Figure 6: Pseudocode of the VNS algorithm.

performed before starting the local search from a new random feasible solution; the number k of local search rounds to be performed before a perturbation occurs; the perturbation size p, that is, the number of flights to be either de-accommodated or randomly shifted in the current best solution to generate the staring solution for a new local search call.

Also the VNS algorithm (Figure 6) starts from a randomly generated feasible solution and explores iteratively the solution neighborhood using one of the two local search. Differently from ILS, it escapes from the basin of attraction of local optima by progressively increasing the size of the neighborhood considered in the local search. When the local search procedures are not able to improve the current solution, the algorithm restarts from a new randomly generated solution where the size of the searched neighborhood is again set to its minimum value. The main idea at the basis of this metaheuristic is extensively exploring a region of the space before migrating to a different one. It includes four parameters: the number r of iterations without an improvement, in terms of number of flights accommodated, that can be performed before restarting the

local search from a new random feasible solution; the number k of local search rounds to be performed before increasing the size of the searched neighborhood; the initial size q and the step size i of each size increase/decrease the searched neighborhood. This last value is expressed in terms of the number of flights that are either de-accommodated in the flight-local-search or randomly shifted in the cost-local-search procedures.

5 Benchmark algorithms

For assessing the performance of TILP, ILS and VNS, we consider two benchmark algorithms. The first one, named CURRENT, tries to mimic the current primary slot allocation process. The second one, the Random Restart Local Search (RRLS), as already mentioned in the Introduction, exploits the same local search procedures as ILS and VNS in a naive way.

CURRENT is a two-step procedure. First, it allocates slots to airlines at each airport separately: for each airport $j \in K$, CURRENT considers only flights with either origin or destination at j and then solves a simplified version of the integer linear programming model of Section 3, which just includes the objective function (4) and airport time and capacity constraints (5)-(7). At the end we obtain the number of slots allocated to each airline a at airport j per time interval t:

$$s_{a,j,t} = \sum_{f \in F_a} w^f_{j,t} \qquad \forall a \in A,\, j \in K,\, t \in T.$$

However, an airline may not be in the position to perform all its flights because it may not have all the necessary departure and arrival slots for its flights. Hence, in its second step, CURRENT computes how many flights each airline can actually accommodate once the slot allocation is over, i.e., as a function of the optimal $s_{a,j,t}$. This is what is currently done separately by each airline before the IATA conference. To simulate this process, we solve a further variation of the integer linear programming model of Section 3. We remove constraints (8) and (9) since currently sector capacity is not taken into account until the very last days; we replace constraints (7) with constraints (16) for imposing the respect of the obtained slot allocation, substituting in this way the competition for capacity with the exploitation of the acquired rights:

$$\sum_{f \in F_a : dest_f = j \vee orig_f = j} w^f_{j,t} - w^f_{j,t-1} \leq s_{a,j,t} \qquad \forall a \in A,\, j \in K,\, t \in T. \qquad (16)$$

The comparison with CURRENT allows us to determine whether the possibly sub-optimal solution to the SSAP found by our algorithms is still more efficient than the current slot allocation process in which all airports are considered separately. In this framework, we measure the efficiency in terms of number of flights that are accommodated after the primary allocation, knowing that

```
S* = ∅
while (time left) do
  S=randomly drawn solution
  if (not all flights are accommodated)
    for (round in 1:k)
      flight-local-search starting form S
  else
    for (round in 1:k)
      cost-local-search starting form S
  if S is better than S* then S* = S
return S*
```

Figure 7: Pseudocode of the RRLS algorithm.

accommodating the remaining ones will be an issue to be solved at the IATA conference.

RRLS is a straightforward random restart local search algorithm, relying on the same local search procedures reported in Section 4. It is shortly described in Figure 7. It has only parameter k, that is the number of rounds of local search to be performed before each restart from a random position. The comparison with this benchmark allows understanding the performance of the metaheuristics somehow independently of the local search procedures.

6 Experimental setup

We have performed an experimental analysis to compare the results achieved by ILS and VNS algorithm with the ones achieved by both the benchmark algorithms and by the TILP.

We have based our comparisons on randomly generated instances. In these instances, we simulate a network with a hub-and-spoke structure: all flights either take-off or land at a hub. We have assumed hubs with limited capacity, whereas we have maintained that spokes have always excess capacity with respect to the demand. We have assumed that also all the sectors have limited capacity. Sectors have been represented as a grid of square cells, and the location of airports has been randomly distributed, imposing a minimum distance of three cells between each pair of airports (Pellegrini *et al.* , 2011). The time intervals in which we have subdivided the time horizon last ten minutes. We have allowed a maximum (backward or forward) shift of three time intervals for each flight, possibly reducing this value when the limit of the time horizon imposes to do so.

We have set the number of flights to accommodate in each instance as a fixed percentage (85%) of the total number of slots available at hubs. In this way, we have generated instances that may vary in size without varying in complexity due to hubs congestion. For each flight, we have randomly drawn the origin

Table 1: Sets of instances tackled. Two values between square brackets $[a, b]$ indicates that a value have been uniformly randomly drawn within such interval. The number of flights is approximated, since it depends on the specific realization of hub capacities; the number reported is the value obtained by considering the average capacity for each hub.

name	small	medium	large
number of hubs	3	5	13
number of spokes	17	15	17
number of sectors	200	200	300
sector capacity	$[25, 30]$	$[25, 30]$	$[25, 30]$
number of time intervals	18	18	120
hub capacity	$[12, 16]$	$[12, 16]$	$[15, 20]$
number of flights	~ 643	~ 1071	~ 29835

and destination airports provided that the following condition are respected: at least one of them should have been a hub; the distances between these two airports distance should not have been greater than twelve time intervals along the shortest route. In all the experiment we considered 50 airlines. Each flight has been randomly associated to one of these airline. The capacity of each hub and each sector has been randomly drawn according to the uniform distributions described in Table 1. The sector and airport capacities indicated in Table 1 refer to the number of allowed movements in one time interval (ten minutes in this experiment).

Finally, we have set the parameters of the cost coefficients in the objective function as follows: we have randomly drawn c^f_{dur} between 20 and 30 and α_f between 20 and c^f_{dur}. In addition, we have imposed $\beta_f = 1.5$ for all $f \in F$. We have set $M = 1000000$.

We have tackled three sets of instances, whose characteristics are listed in Table 1. We have selected them for testing the algorithms under three experimental conditions:

1. *small instances* that can be solved by TILP in 360 seconds;

2. *medium instances* that cannot be solved by TILP in 600 seconds. However, by letting the TILP run an arbitrary amount of time, we were able to prove that the metaheuristic algorithms could find the optimal solution;

3. *large instances* that cannot be solved by TILP in any case. They represent instances of realistic size, taking into account that, in Europe, there are about 30000 flights a day in the high season.

Remark that the difference between small and medium instances is the number of hubs and spokes, which imply a different number of flights: an increase of the number of hubs from three to five leads to an increase of the total expected

Table 2: Settings tested and selected in the tuning.

parameter	setting tested	ILS selected	VNS selected	RRLS selected
t	1, 10, 50, 100, 200	200	10	200
q	20, 40, 60, 80, 100	20	20	50
r	5, 10, 25, 50, 100	10	50	
k	1, 5, 10, 25, 50	50	5	50
p	100, 200, 400, 600	100		
i	5, 10, 15, 25, 50, 100		25	

number of slots from 42 to 70 per time interval, and thus, in 18 time intervals, to an increase of the total expected number of flights from 643 to 1071. In the large instances, both the network, the number of hubs and the number of time intervals increase, so that the number of flights grows quite strongly. We have solved 30 instances of each set, allowing 360 seconds of computation on small instances for each algorithm that we have described but CURRENT, for which we do not impose any time restriction. The corresponding computation time for medium and large instances is 600 and 3600 seconds, respectively. Following Birattari (2004), we have performed a single run for each instance with each algorithm. For TILP and CURRENT we have used XPRESS optimizer version 20.00.11, setting an optimality gap of 0.0001%. For ILS, VNS, RRLS we have used the $c++$ compiler with *gcc* version 4.4.3. We have run all the experiments on a Intel XEON with 16 CPUs at 2.27 GHz and with 16.00Gb of Ram running Linux.

For selecting the setting of the parameter of ILS, VNS and RRLS we have used the I-Frace (López-Ibáñez *et al.* , 2011) procedure. I-Frace is an automatic tuning procedure that progressively samples the space of parameter settings and discards dominated settings. For each algorithm, we have allowed I-Frace to perform 1000 ten-minute runs on ten small instances, that have not been then used in the experimental analysis. Table 2 reports the settings tested and the ones selected by I-Frace. We reported the meaning of parameters in Section 4 for ILS and VNS, and in Section 5 for RRLS. We have used these settings for all the experiments. The results achieved by the metaheuristic algorithms with the settings returned by I-Frace are very good for all three sets of instances, i.e., small, medium and large instances. Thus, we have not felt it necessary to perform a specific tuning for each set.

7 Experimental results

In this section, we report the results of the performance comparison of TILP, ILS and VNS, using as benchmark algorithms, CURRENT and RRLS as described in Section 5. We address small, medium and large instances. For each instance,

the comparison is based on four criteria:

1. the percentage of accommodated flights with respect to the total number of flights of the instance. A more precise indicator of the algorithms' performance would be the percentage of accommodated flights with respect to the maximum number of flights that can actually be accommodated. In fact, the different constraints (5)-(15) may not allow to accommodate all the instance flights. However, for the set of large instances we are unable to compute the maximum number of flights that can be accommodated whereas for the small and medium instances we see that all instance flights are accommodated. Thus we use the number of the instance flights as upper bound for the maximum number of flights that can be accommodated.

2. the percentage error in terms of the overall objective function, that is,

$$e\% = \frac{\hat{z} - z^*}{z^*}$$

   where $\hat{z}$ is the overall cost associated to the slot allocation returned by the tested algorithm and z^* is the optimal value of the objective function (4). The error $e\%$ is reported only for those instances for which we have been able to compute the value of z^*.

3. if all the instance flights are accommodated or not.

4. the percentage error in terms of total shift cost, that is,

$$e_f\% = \frac{\hat{C}_f - C_f^*}{C_f^*}$$

   where $\hat{C}_f$ is the shift cost associated to the slot allocation returned by the tested algorithm and C_f^* is the shift cost component of the optimal value z^* of the objective function (4). Again, this error $e_f\%$ is reported only for those instances for which we have been able to compute the value of z^*.

Table 3 summarizes the findings. The first, second and fourth column shows the average value (over all instances of the corresponding set) of the first, second and fourth performance criterion, respectively. The third column is associated to the third performance criterion and indicates in how many instances (out of 30) all flights have been accommodated.

In some cases the exact solver recognized as optimal solution an actually suboptimal one. This is due to the fact that the total shift cost is the secondary objective, and thus it is weighted some orders of magnitude less than the primary objective function, that is, the number of accommodated flights. In these suboptimal cases, the optimizer accepts a difference of some units in terms of total shift cost, due to the presence of the tolerated optimality gap of 0.0001%. Then a negative percentage error in terms of total shift cost may occur, when we compare these results with the metaheuristic ones.

Table 3: Summary of the results obtained in the experimental analysis: mean percentage number of accommodated flights (% FLIGHTS), mean percentage error in terms of the overall objective function (% OBJ. FUN.), number of instances in which the optimal number of flights is accommodated (INST.) and mean percentage error in terms of the total shift cost in the instances in which the optimal number of flights are accommodate (% COST).

SETS OF INSTANCES	ALGORITHMS	% FLIGHTS	% OBJ. FUN.	INST.	% COST
small instances	TILP	96.667	3.333	29	0.076
	ILS	100.000	0.000	30	0.430
	VNS	100.000	0.000	30	−0.206
	RRLS	100.000	0.000	30	0.088
	CURRENT	91.069	8.930	0	–
medium instances	TILP	36.189	63.811	3	0.315
	ILS	100.000	0.000	30	0.546
	VNS	100.000	0.000	30	−0.016
	RRLS	100.000	0.000	30	0.254
	CURRENT	94.801	5.199	0	–
large instances	TILP	0.000	–	0	–
	ILS	99.952	–	1	–
	VNS	99.894	–	0	–
	RRLS	99.910	–	0	–
	CURRENT	0.000	–	0	–

We see that ILS, VNS and RRLS could solve all sets of instances. TILP and CURRENT, instead, could not deal with large instances. In both cases, the issue was loading the matrix of the linear programming model: for TILP, at the beginning of the computation; for CURRENT, after solving the single airport sub-problems, for assessing the number of flights that may be performed based on the allocated slots. Thus, for large instances we do not report any result for TILP and CURRENT. In these cases, as we were not able to compute the optimal solution, we do not present any result in terms of percentage error.

In small and medium instances, ILS, VNS and RRLS accommodate all flights in all instances. This is not the case either for TILP, that fails in one small and twenty-seven medium instances, or for CURRENT, that fails in all instances. These failures emerge also in the mean percentage error in terms of overall objective function value: while this error is zero for ILS, VNS and RRLS, it is strictly positive for TILP and CURRENT. TILP and CURRENT are, in fact, significantly worse than the other three algorithms according to the t-test with 95% confidence interval. Figure 8 reports the boxplots of the distribution of the percentage error in terms of total shift cost made by the algorithms on small and medium instances. It deals only with the instances for which the algorithms find a solution accommodating all flights. If this is not the

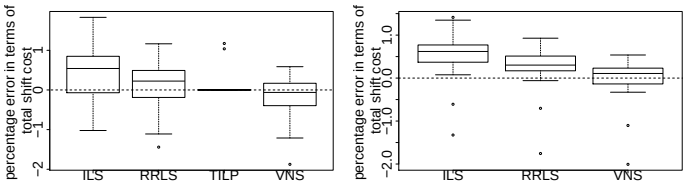


Figure 8: Distribution of the percentage error made by TILP, ILS, VNS and RRLS on small and medium instances.

case, the difference in terms of total shift cost is not relevant according to the definition of the SSAP. On small instances (Figure 8, left), TILP almost always achieves the optimal solution in terms of total shift cost. ILS appears to be the worst performing algorithm, while VNS is the best. The figure does not include CURRENT, since for no instance it accommodated all flights. On medium instances (Figure 8, right), the same relation holds for what concerns ILS, VNS and RRLS. The figure does not include TILP, since it accommodated all flights in only three instances, and thus the definition of a distribution of the errors would be meaningless. The relation among ILS, VNS and RRLS, with ILS being the worst, VNS the best, and RRLS being between them, is reverted in large instances: as it can be seen in Table 3, here ILS outperforms its competitors, while VNS ranks last. Also in this case, the differences are statistically significant according to the t-test with 95% confidence interval.

Figures 9, 10 and 11 report the results achieved by the algorithms throughout the run on small, medium and large instances, respectively. We do not consider CURRENT in this analysis, since it is not comparable to the other algorithms in terms of computational time: the single airport sub-problems may be solved in parallel in potentially different locations. For small and medium instances, we report only the results on one representative instance per set. For large instances, we show three instances, since we consider only one criterion for evaluating the performance, as discussed at the beginning of this section. The results on all the other instances are available in the Appendix. When we report the percentage error in terms of total shift cost (Figures 9 and 10, bottom), we show the result achieved by the algorithms only as far as they find a solution accommodating all flights, which may occur rather late in their runs. In small and medium instances (Figures 9 and 10), ILS, VNS and RRLS accommodate all flights in about one second, and then spend the remaining computation time for improving the solution in terms of total shift cost. This leads to very small percentage errors. TILP, instead, needs quite a long time for accommodating all flights, but, as soon as it does, it finds the optimal solution in terms of the total shift cost. In large instances (Figure 11), ILS, VNS and RRLS need very few seconds for accommodating almost all flights, about 99.8% of them, or more. Then, they keep accommodating flights at a slower rate throughout the

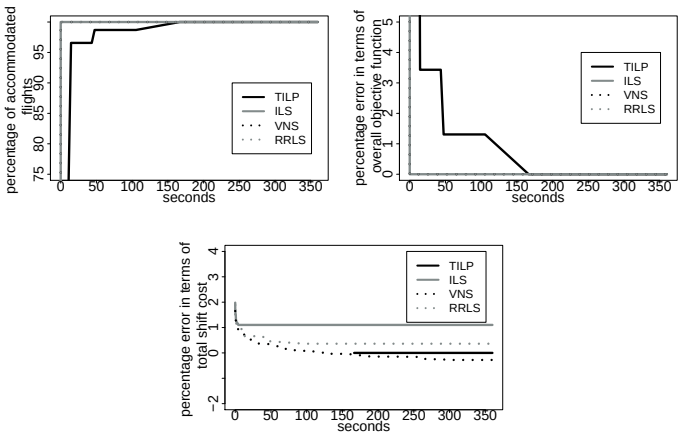


Figure 9: Results achieved by TILP, ILS, VNS and RRLS throughout the run on one small instance.

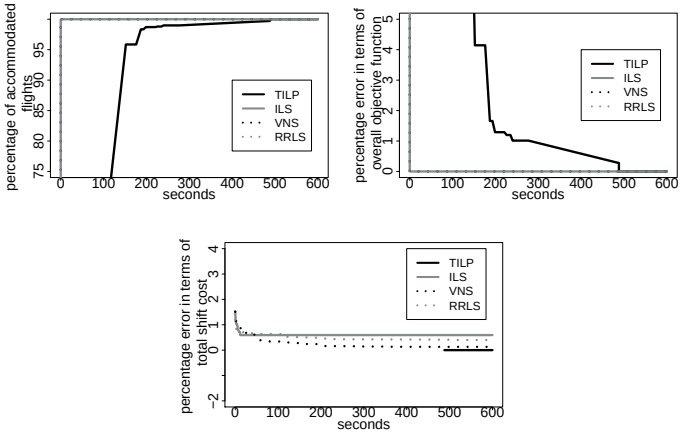


Figure 10: Results achieved by TILP, ILS, VNS and RRLS throughout the run on one medium instance.

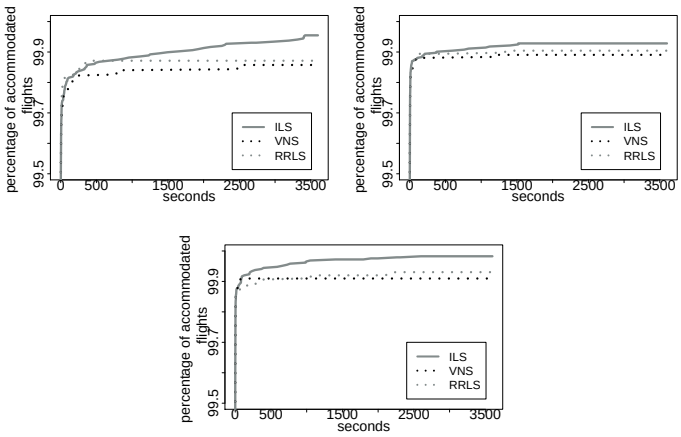


Figure 11: Results achieved by TILP, ILS, VNS and RRLS throughout the run on three large instances.

remaining computation time. In all the instances, the number of accommodated flights increases until the end of the run.

8 Conclusions

In this paper, we formally introduced the simultaneous slot allocation problem: it requires to allocate slots to airlines taking into account simultaneously the capacities of all the airports and of all the sectors, so that to ensure feasible routes.

We have introduced and compared three algorithms for tackling the SSAP: a truncated integer linear programming model named TILP, an iterated local search named ILS and a variable neighborhood search named VNS. We have assessed their performances on three sets of randomly generated instances of different size, and we have observed their behavior with respect to two benchmark algorithms: CURRENT and RRLS.

Both ILS and VNS outperform CURRENT on all sets of instances, while TILP does so only on small ones, due to memory consumption and computation time requirement issues. Differently from TILP, ILS and VNS are able to tackle also instances including about 30000 flights, that is, about the number of flights that are performed in Europe in one high season day. For what concerns RRLS, its performance appear always worse than either ILS or VNS, and better than the other one: for small and medium instances VNS is the best performing; for large instances ILS is the best. Yet, due to the inversion in the relative performance of ILS and VNS, we cannot discard RRLS from the set of well performing algorithms. Thus, the local search procedures we proposed appear

very effective: even by applying them in a naive random restart way, their performance are quite positive. Still, by combining them with a more thorough exploration of the space as it is done in metaheuristics, one may achieve very high quality performance.

In summary, metaheuristics appear valid approaches for tackling the SSAP, even in realistic-size instances. In future research, we will extend the definition of the SSAP to include the warrant of grandfather rights.

Acknowledgements

The work of Paola Pellegrini is funded by a Bourse d'excellence Wallonie-Bruxelles International.

References

Airport Council International Europe. 2004. *Study on the use of airport capacity.* Bruxelles, Belgium.

Airport Council International Europe. 2009. *ACI Europe position on the proposed revision of the Council Regulation (EEC) No 95/93 on common rules for the allocation of slots at Community airports.* Presentation on the TRAN Meeting at the European Parliament, March 25, Strasbourg, France.

Barnhart, C., Belobaba, P., & Odoni, A. 2003. Applications of operations research in the air transport industry. *Transportation science*, **37**(4), 368–391.

Bertsimas, D., Lulli, G., & Odoni, A. 2011. An integer optimization approach to large-scale air traffic flow management. *Operations research*, **59**(1), 211–227.

Birattari, M. 2004. *On the estimation of the expected performance of a metaheuristic on a class of instances. How many instances, how many runs?* Tech. rept. TR/IRIDIA/2004-01. IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Brooker, P. 2008. SESAR and NextGen: Investing In New Paradigms. *Journal of navigation*, **61**, 195–208.

Castelli, L., Pellegrini, P., & Pesenti, R. 2010. Airport slot allocation in europe: economic efficiency and fairness. *International journal of revenue management.*

de Wit, J., & Burghouwt, G. 2007. *The impact of secondary slot trading at amsterdam airport schiphol.* http://www.seo.nl/binaries/publicaties/rapporten/2007/957.pdf.

DotEcon Ltd. 2001. *Auctioning airport slots.* http://www.dotecon.com/publications/slotauctr.pdf.

DotEcon Ltd. 2006. *Alternative allocation mechanisms for slots created by new airport capacity.* http://www.dft.gov.uk/pgr/aviation/airports/alternativeallocationmechani.pdf.

Eurocontrol Experimental Centre. 2008. *Long-Term Forecast: IFR Flight Movements 2008-2030, v1.0.* Forecast prepared as part of the Challenges of Growth 2008 project, Brussels, Belgium.

European Commission. 1993. *Council Regulation (EEC) No 95/93 of 18 January 1993 on common rules for the allocation of slots at Community airports.*

European Commission. 2002. *Regulation (EC) No 894/2002 of the European Parliament and of the Council of 27 May 2002 amending Council Regulation (EEC) No 95/93 on common rules for the allocation of slots at Community airports.*

European Commission. 2003. *Regulation (EC) No 1554/2003 of the European Parliament and of the Council of 22 July 2003 amending Council Regulation (EEC) No 95/93 on common rules for the allocation of slots at Community airports.*

European Commission. 2004. *Regulation (EC) No 793/2004 of the European Parliament and of the Council of 21 April 2004 amending Council Regulation (EEC) No 95/93 on common rules for the allocation of slots at Community airports.*

European Commission. 2007. *Communication from the Commission Communication on the application of Regulation (EC) 793/2004 on common rules for the allocation of slots at Community airports [COM(2007)704].*

European Commission. 2008. *Communication from the Commission on the application of Regulation (EEC) No 95/93 on common rules for the allocation of slots at Community airports [COM(2008)227].*

European Commission. 2009. *Regulation (EC) No 545/2009 of the European Parliament and of the Council of 18 June 2009 amending Regulation (EEC) No 95/93 on common rules for the allocation of slots at Community airports.*

European Commission. 2010. *Commission Staff working paper on deployment of the Single European Sky technological pillar (SESAR).*

Fonlupt, C., Robillard, D., Preux, P., & Talbi, E-G. 1999. *Fitness landscape and performance of metaheuristics.* Kluwer Academic Press, USA. Pages 345–358.

Fukui, H. 2010. An empirical analysis of slot trading in the Unites States. *Transportation research part b*, **44**, 330–357.

Hansen, P., & Mladenovič, N. 2001. Variable neighborhood search: Principles and applications. *European journal of operational research*, **130**(3), 449–467.

Holt, D., Meaney, A., Noble, R., Riley, C., Shaw, A., & Oxera Consulting Ltd. 2007. Assessing the welfare impact of the introduction of secondary slot trading at community airports. *In: European transport conference 2007.*

Hoos, H. H., & Stützle, T. 2004. *Stochastic local search. foundations and applications.* San Francisco, CA, USA: Morgan Kaufmann Publishers.

IATA. 2011. *Worldwide scheduling giudelines, 21th edition.* Montreal, Canada.

Kleit, A.N., & Kobayashi, B.H. 1996. Market failure or market efficiency? Evidence on airport slot usage. *Research in transportation economics*, **4**(1), 1–32.

Kösters, D. 2007. *Study on the usage of declared capacity at major German airports.* http://www.eurocontrol.int/prc/gallery/content/public/Docs/ FINAL_REPORT_Study_Capacity_Usage.pdf.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., & Birattari, M. 2011. *The `irace` package, iterated race for automatic algorithm configuration.* Tech. rept. 2011-04. IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Lourenço, H.R., Martin, O., & Stützle, T. 2003. *Iterated local search.* Kluwer Academic Publishers. Pages 321–353.

Maldoom, D. 2003. Autioning capacity at airports. *Utilities policy*, **11**, 47–51.

Mott MacDonald Limited. 2006. *Study on the impact of the introduction of secondary trading at community airports.* http://www.euaca.org/documents/2006_slots_final_report.pdf_211108_054651.pdf.

NERA Economic Consulting. 2004. *Study to assess the effects of different slot allocation schemes.* http://www.nera.com/67_4921.htm.

Pellegrini, P., Castelli, L., & Pesenti, R. 2011. Secondary trading of airport slots as a combinatorial exchange. *Transportation research part e: Logistics and transportation review.* To appear.

Rassenti, S.J., Smith, V.L., & Bulfin, R.L. 1982. A combinatorial auction mechanism for airport time slot allocation. *Bell journal of economics*, **13**(2), 402–417.

Sentance, A. 2003. Airport slot auctions: desirable or feasible? *Utilities policy*, **11**, 53–57.

SESAR Consortium. 2008. *Milestone deliverable D4: The ATM deployment sequence.*

Starkie, D. 1998. Allocating airport slots: a role for the market? *Journal of air transport management*, **4**, 111–116.

UK Civil Aviation Authority. 2001. *The implementation of secondary slot trading.* https://www.auc.org.uk/docs/5/ergdocs/slotsnov01.pdf.

Verhoef, E.T. 2010. Congestion pricing, slot sales and slot trading in aviation. *Transportation research part b*, **44**, 320–329.

Appendix

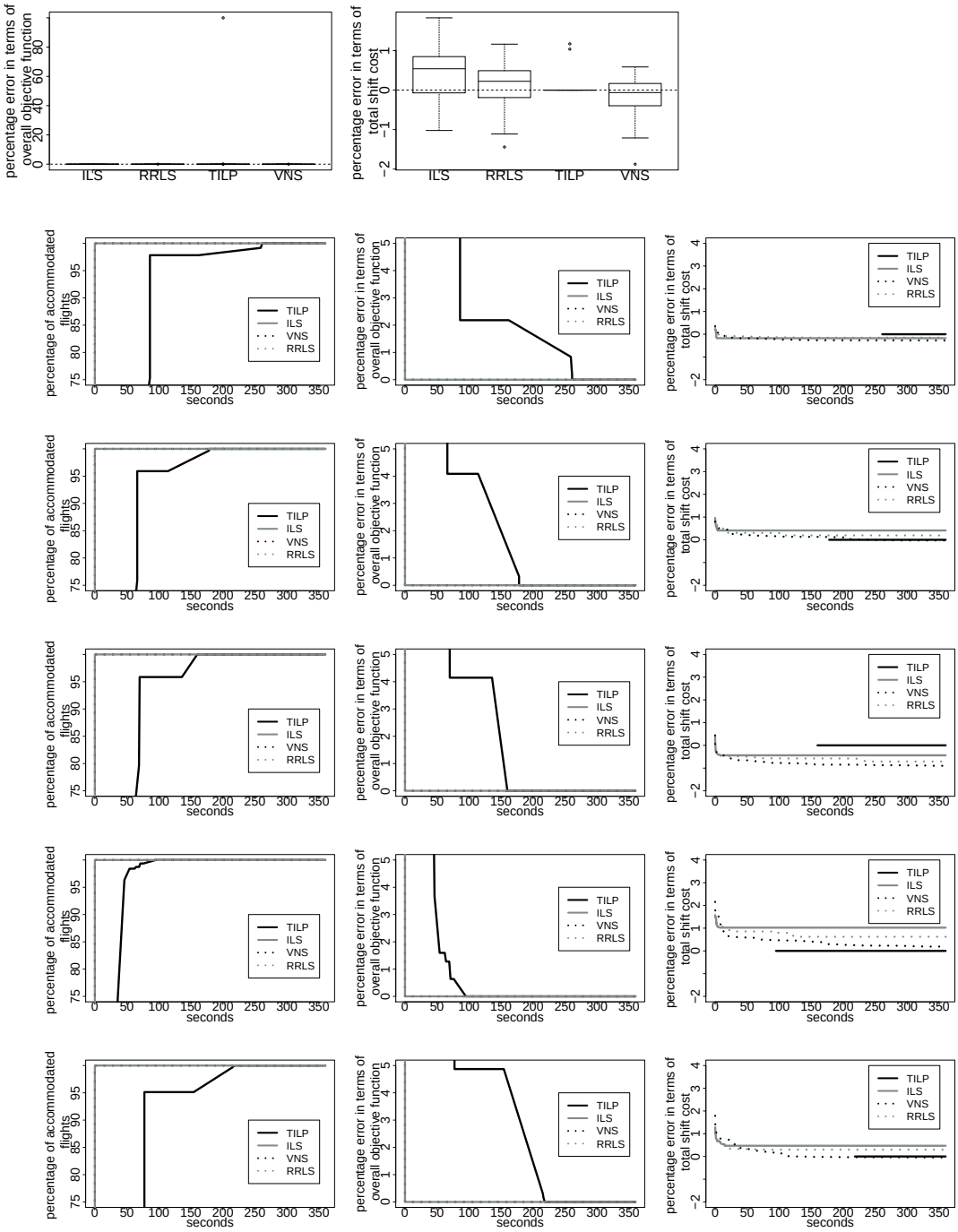
In this appendix, we report the graphs depicting all our experimental results. These graphs are organized according to the size of the instances. For the small and medium instances, we fist presents two boxplots that summarizes the results for all the instances, and then a set of three graphs for each of the instance considered. For the large instances, we just present a graph for each instance as we cannot compare the heuristic results with the optimal ones.

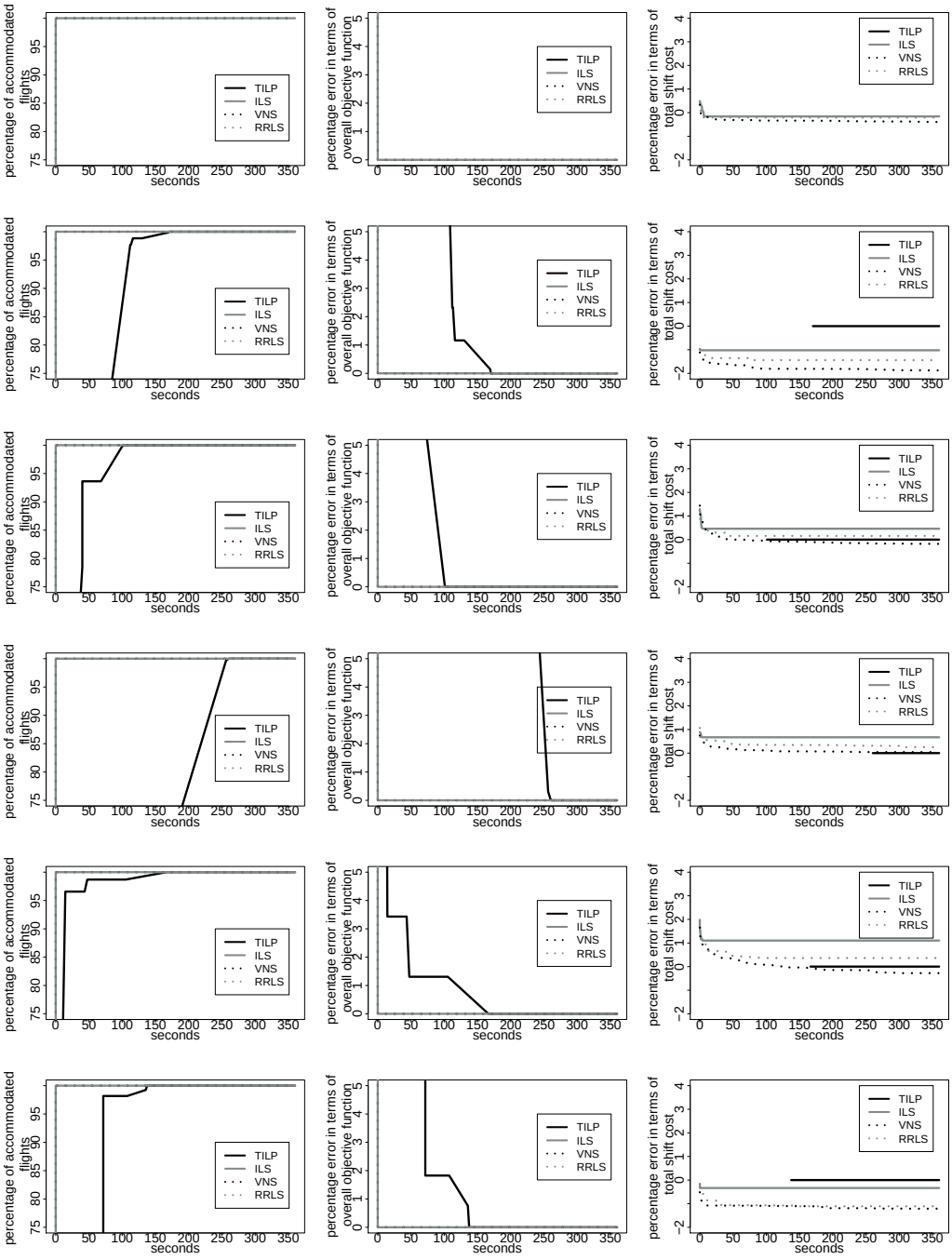
The first boxplot for the small and medium instances shows the percentage error with respect to the optimum in terms of overall objective function value on all the instances tackled. The second bloxplot shows the results in terms of total shift cost on the instances in which the all the flights are accommodated. We found the optimal solutions by solving the model reported in Section 3 with a gap of 0.0001% and no computation time constraints. The presence of the gap on the overall objective function value implies that in some instances, the solver recognizes as optimum a solution that is not such in terms of total shift cost: an error of some units may occur in this case. For this reason, in some instances, metaheuristics find a solution that appears better than the optimum.

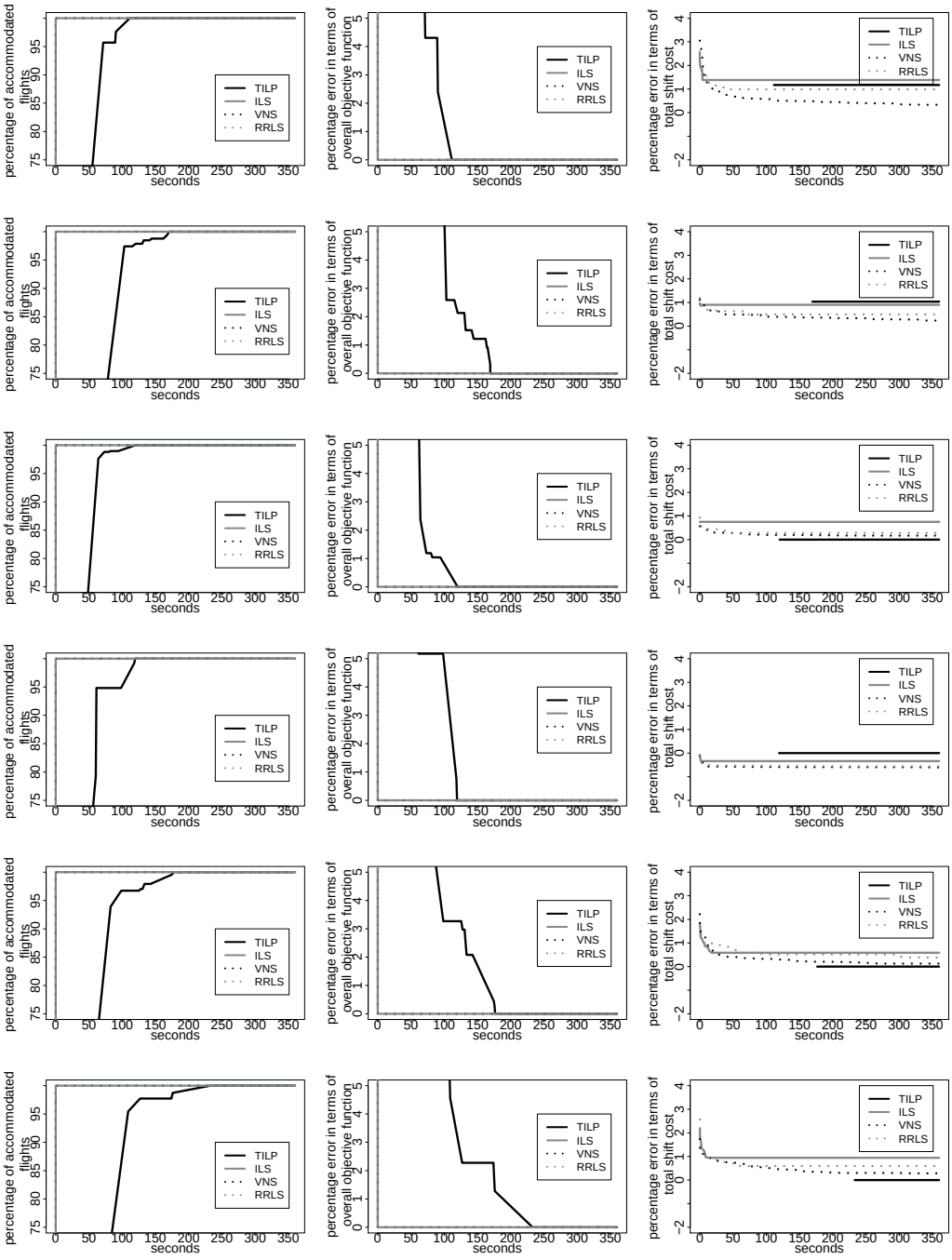
On the other hand, the sets of three graphs for each of the instance considered show the percentage of accommodated flights with respect to the optimum, the percentage error in terms of overall objective function, and the percentage error in terms of total shift cost. In the latter graphs, we consider only solutions in which all flights are accommodated.

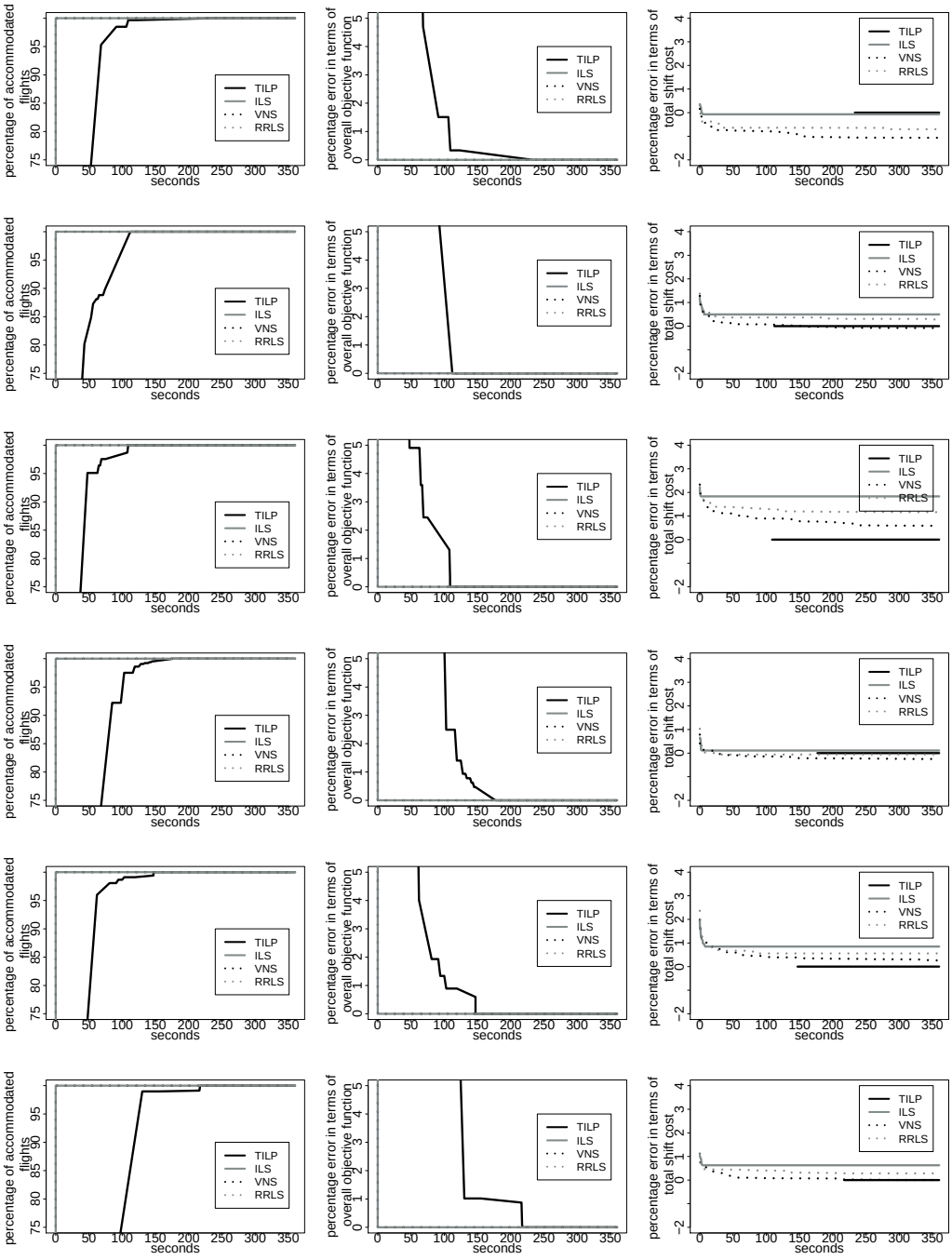
For large instances, a graph for each instance shows the percentage of accommodated flights with respect to the total number of flights in an instance. We do not propose any other comparison since the exact solver cannot load any of these instances, and thus we do not know either the optimal solution or any solution found by TILP. Furthermore, in most of the instances ILS, VNS and RRLS do not accommodate the same number of flights, which makes any comparison in terms of total shift cost meaningless.

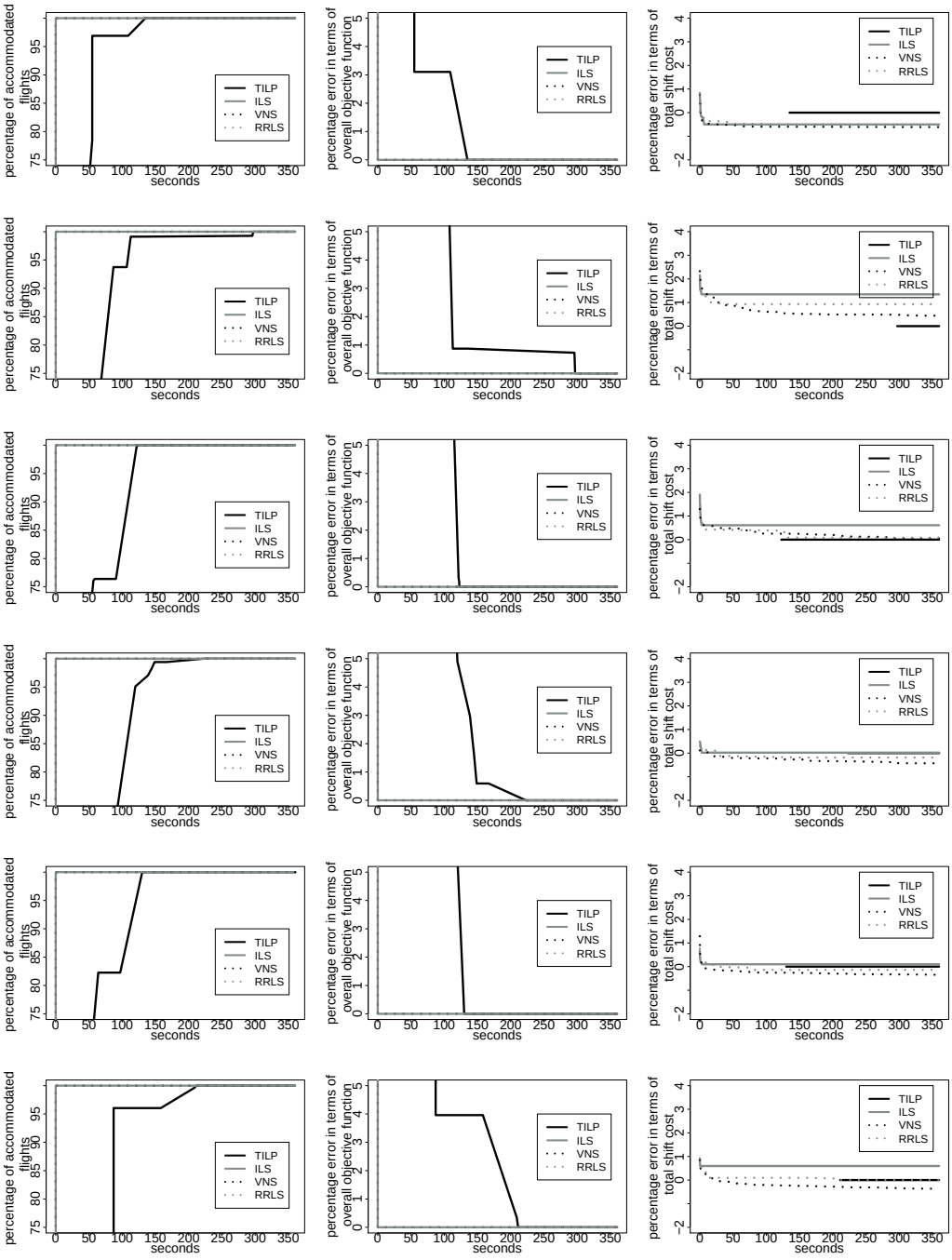
Small instances

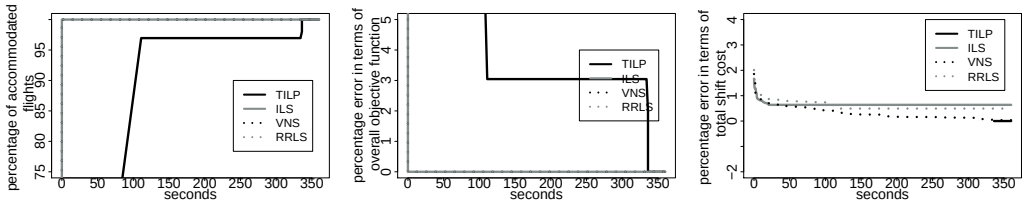



percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

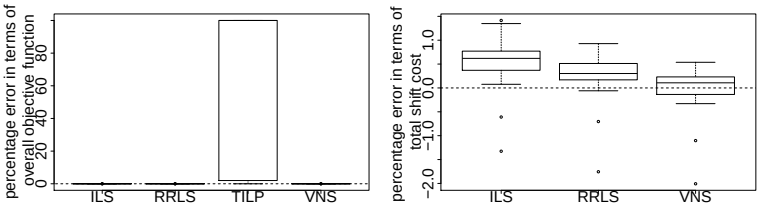

percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

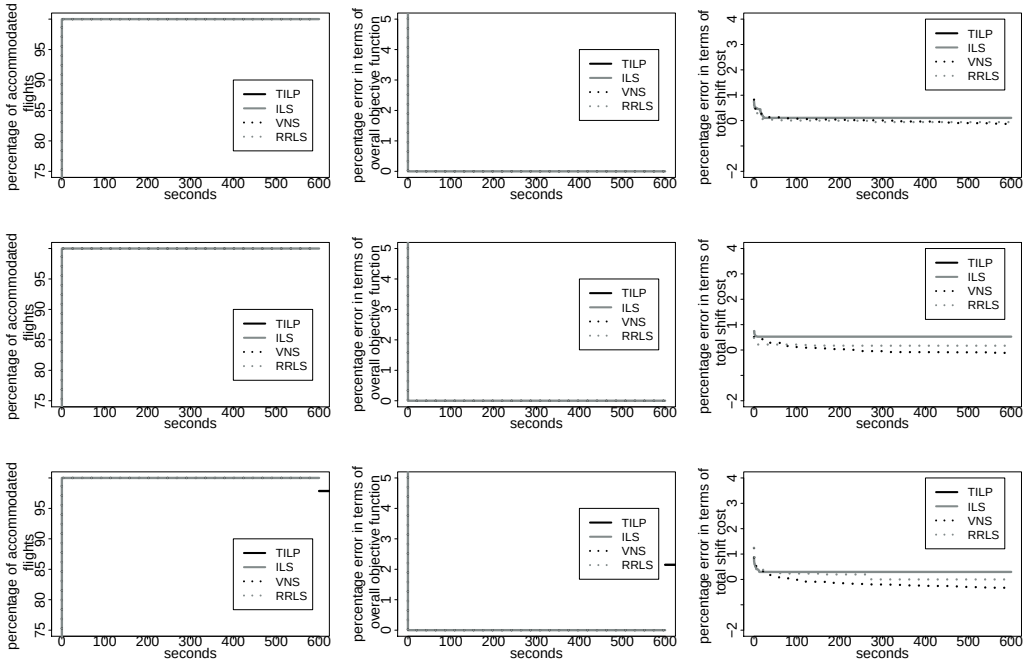

percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

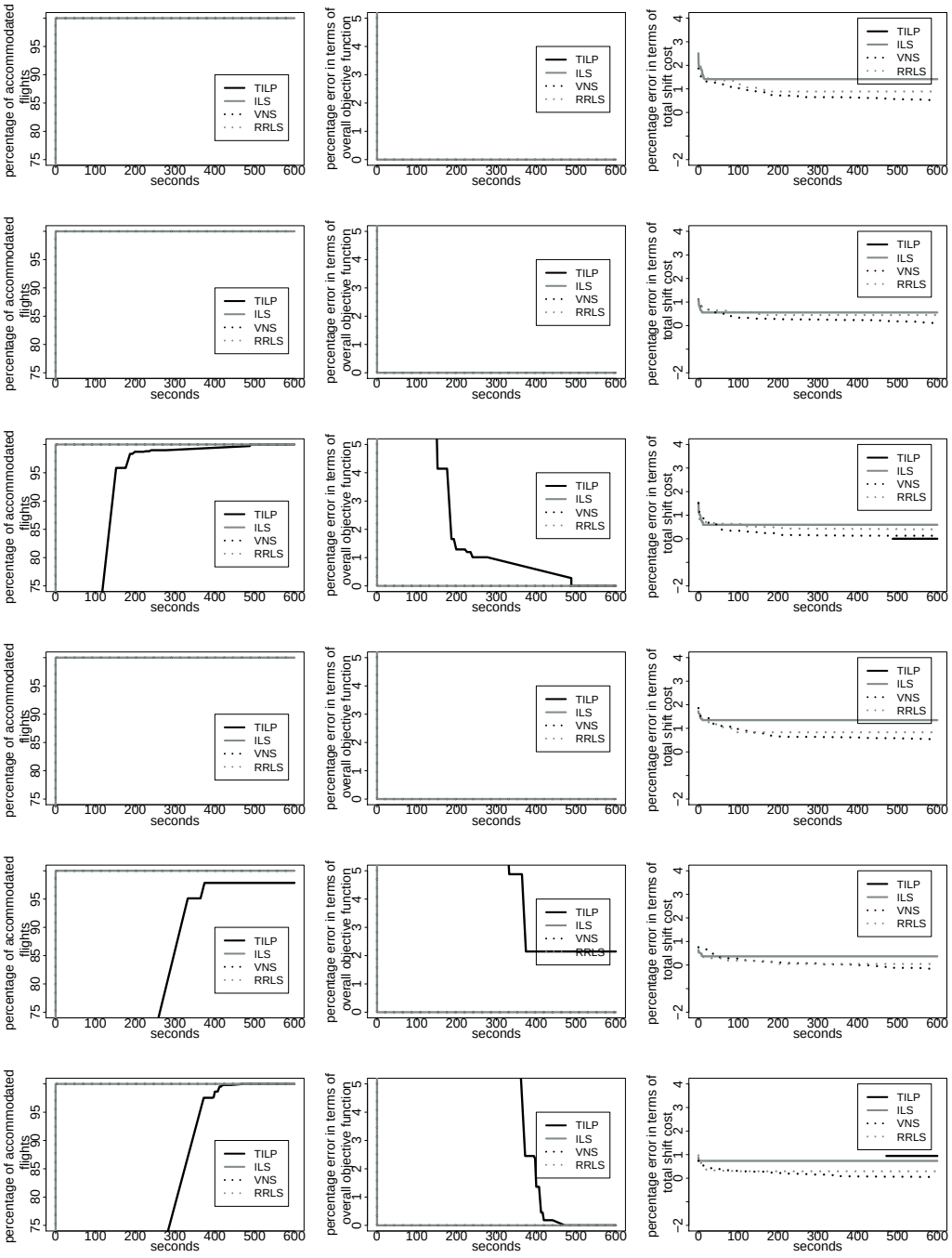

percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

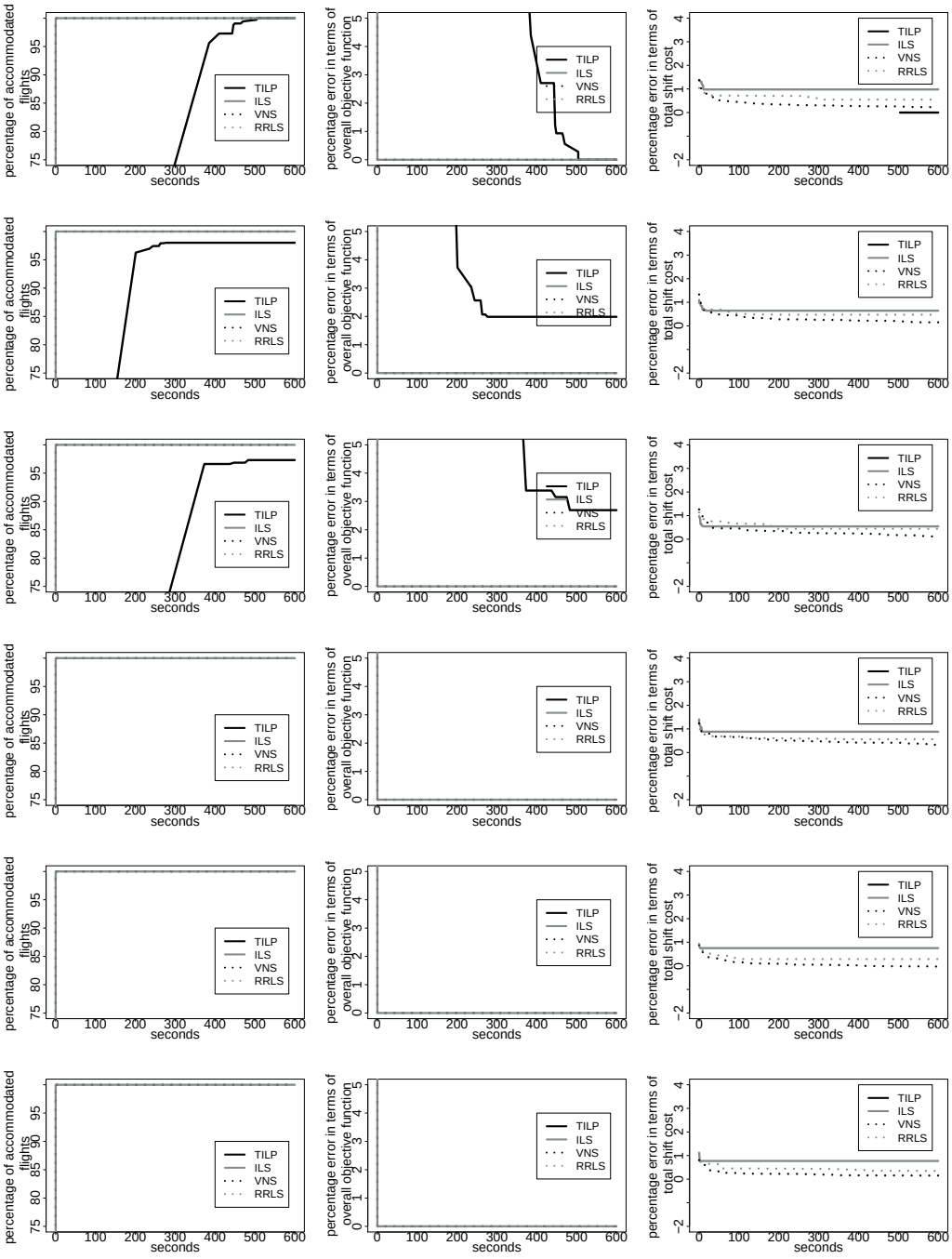


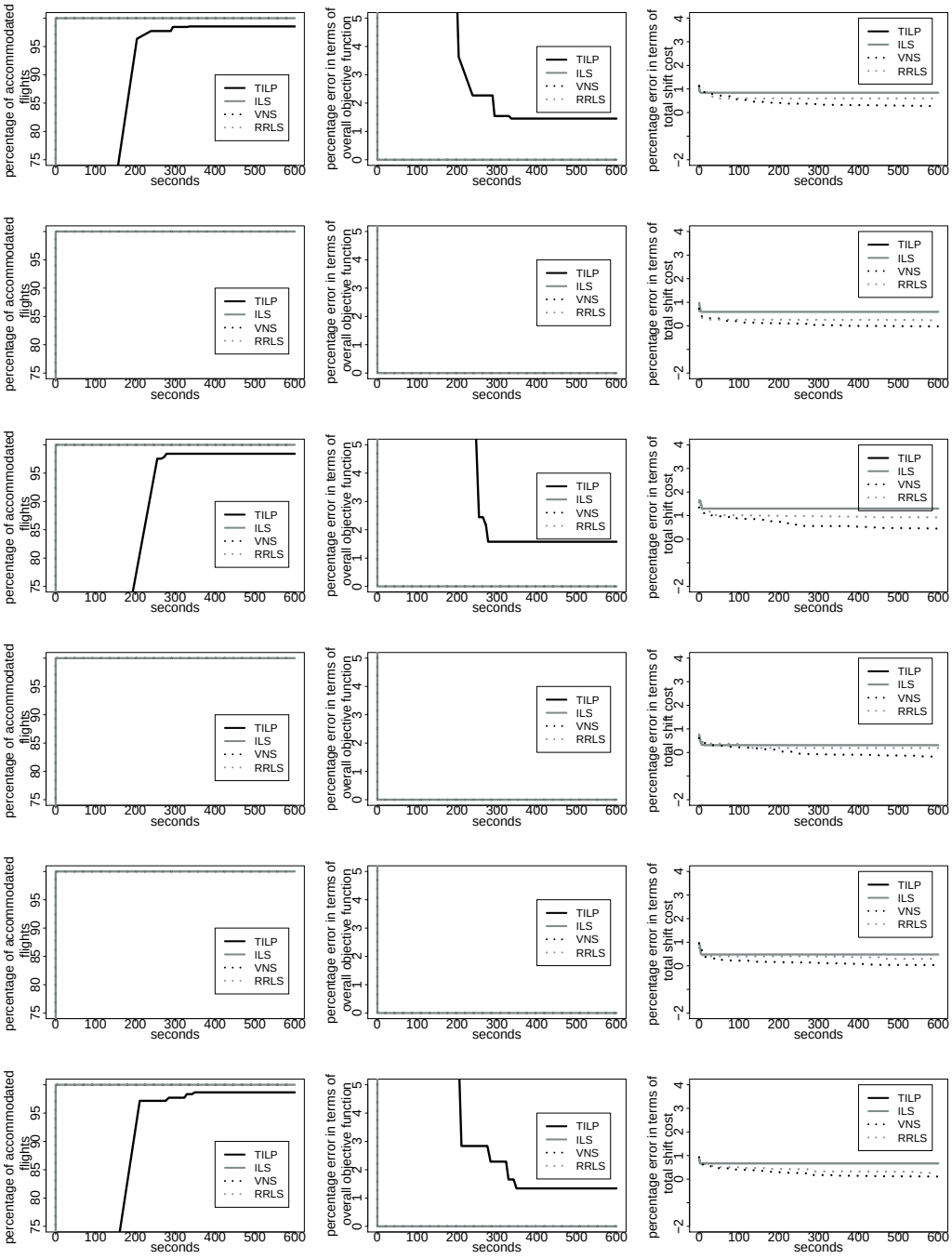
Medium instances

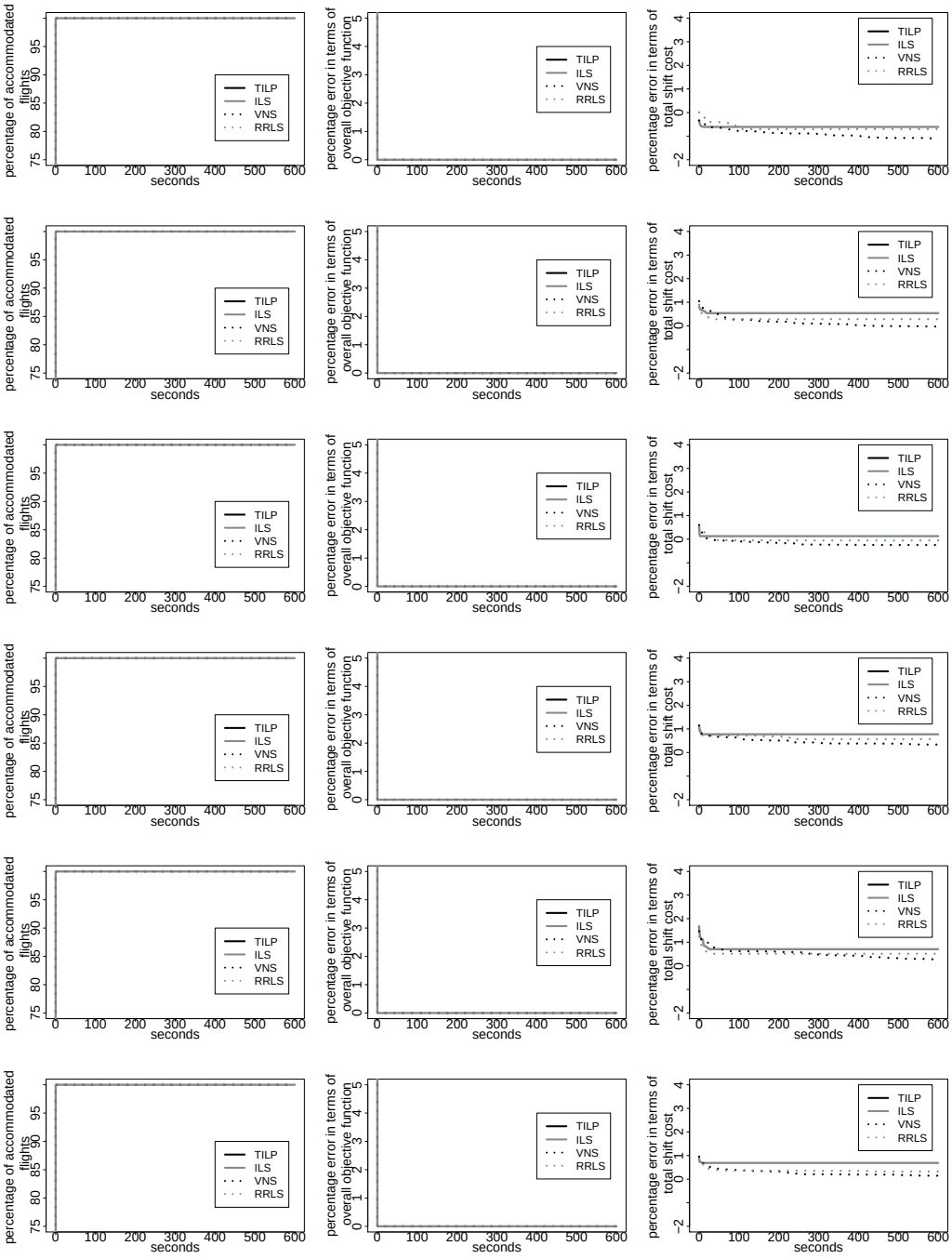


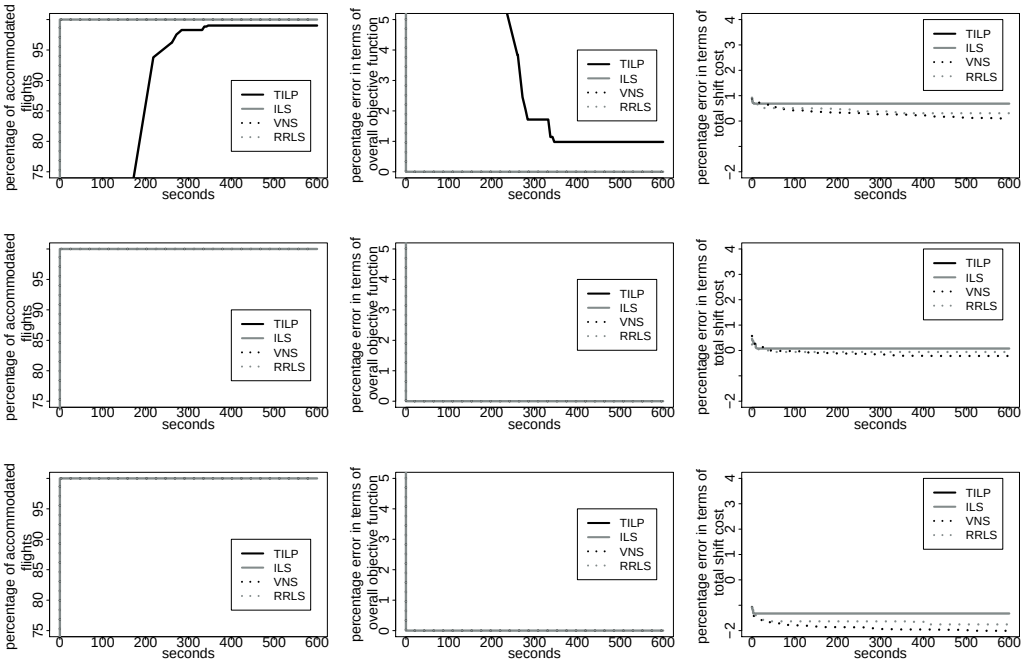



percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

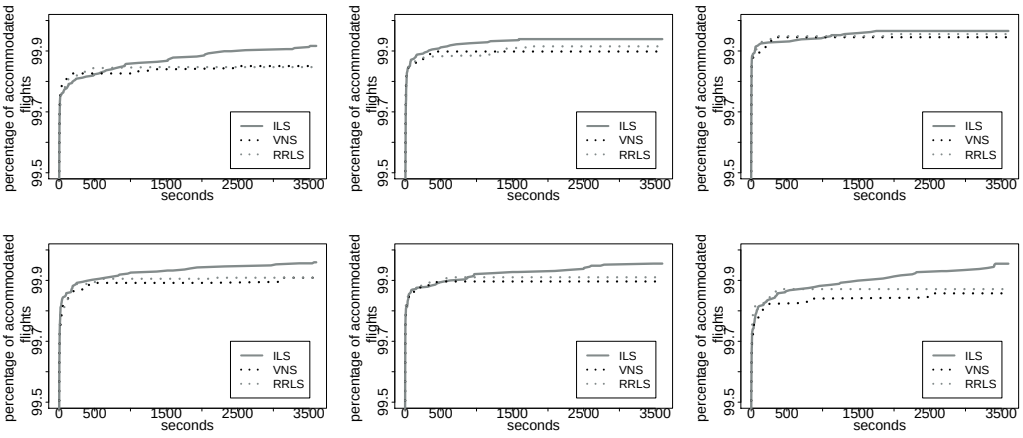

percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

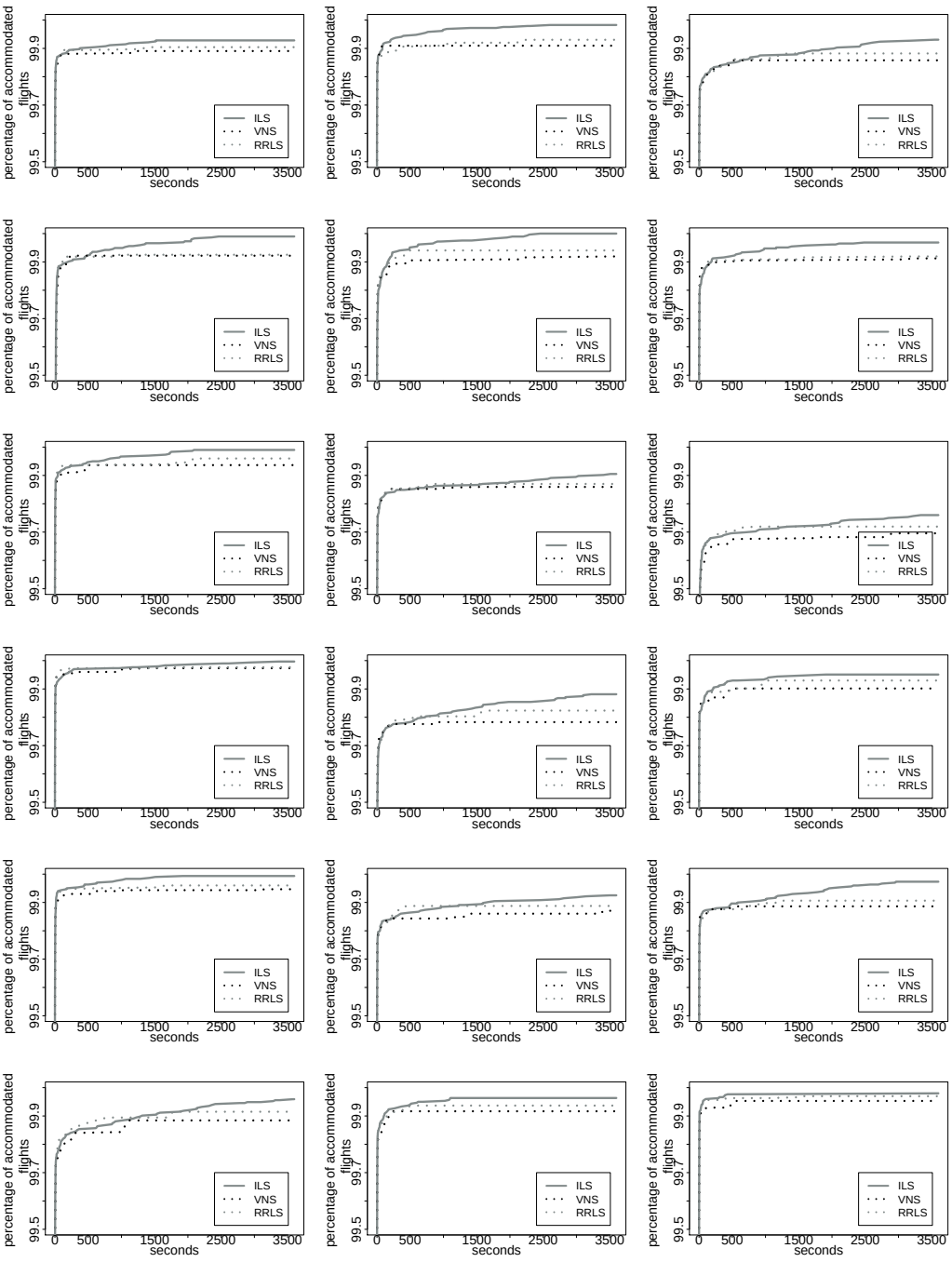

percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS

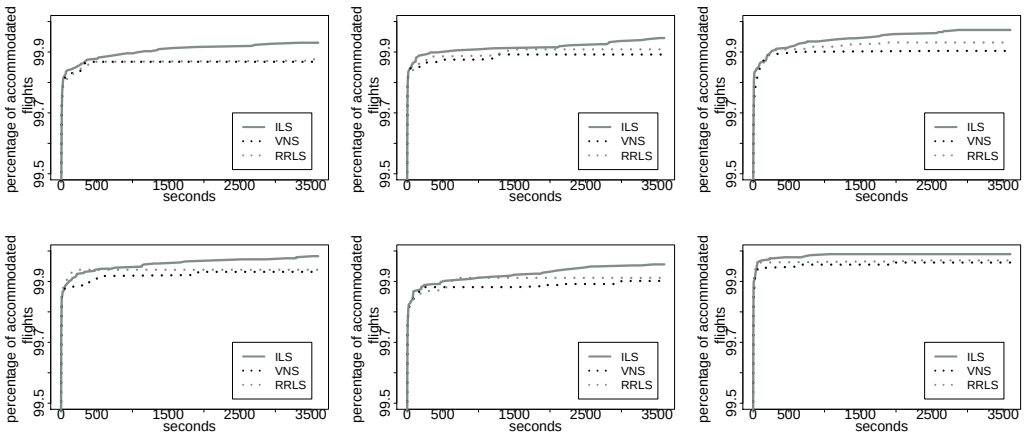

percentage of accommodated flights
percentage error in terms of overall objective function
percentage error in terms of total shift cost
seconds
TILP
ILS
VNS
RRLS



Large instances




percentage of accommodated flights
99.5
99.7
99.9
0
500
1500
2500
3500
seconds
ILS
VNS
RRLS


percentage of accommodated flights
99.5
99.7
99.9
0
500
1500
2500
3500
seconds
ILS
VNS
RRLS