INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 00 (2014) 1–19
DOI: 10.1111/itor.12063

Heuristics and matheuristics for a real-life machine reassignment problem

Ramon Lopes, Vinicius W.C. Morais, Thiago F. Noronha and Vitor A.A. Souza

Department of Computer Science, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG 31270-010, Brazil

E-mail: ramon@dcc.ufmg.br [Lopes]; vwcmorais@dcc.ufmg.br [Morais]; tfn@dcc.ufmg.br [Noronha]; vitor.andrade@dcc.ufmg.br [Souza]

Received 31 December 2012; accepted 1 November 2013

Abstract

This paper addresses a real-life machine reassignment problem proposed in the Google ROADEF/EURO Challenge (2012). In this paper, we propose a linear integer programming (IP) formulation and iterated local search (ILS) heuristics for approximately solving this problem. Different versions of the ILS heuristics are presented. Two of these versions rely on IP-based perturbations, whereas the other two are based on randomized perturbations. We also propose efficient restricted versions of the classic perturbation and local search procedures based on the "shift" and "swap" neighborhoods. Computational experiments showed that the IP-based heuristics are competitive with the best heuristics in the literature.

Keywords: machine reassignment problem; iterated local search; heuristics; linear integer programming

1. Introduction

ROADEF is a traditional biannual operational research competition that proposes a combinatorial optimization problem in collaboration with an industrial partner. The last ROADEF challenge was a collaboration with Google, and the challenge objective was to improve the usage of a set of machines in a data center.

Let M be a set of machines, P a set of processes running on these machines, and R a set of resources (e.g. CPU, RAM memory, etc.) that are provided by machines and consumed by processes running on the machines. Given an initial assignment I of processes to machines, such that $I_p \in M$ denotes the machine on which a process $p \in P$ is initially running, the machine reassignment problem (MRP) studied in this paper consists in finding an alternative process-to-machine reassignment X that optimizes the usage of the machine resources and that satisfies the five constraints described below. We denote by $X(p) \in M$ the machine on which process p runs in the reassignment X.

The "capacity constraints" guarantee that all machines have enough available resources to run the processes assigned to them. Let $C_{mr} \in \mathbb{N}$ be the capacity of a resource $r \in R$ for a machine $m \in M$ and $D_{pr} \in \mathbb{N}$ be the demand of r for a process $p \in P$, the capacity constraints can be formulated as

$$U(m, r) \leq C_{mr}, \quad \forall\, m \in M, r \in R, \tag{1}$$

where $U(m, r) = \sum_{p \in P | X(p) = m} D_{pr}$ is the usage of r for m according to the reassignment X. We avoid the notation $U(m, r, X)$ and write $U(m, r)$ for the sake of a cleaner notation. We use same notations for all the functions that depend on X.

A resource $r \in R$ is said "transient" if it is reserved for a process $p \in P$ simultaneously on machines I_p and $X(p)$. For example, the disk space used by p in I_p can only be freed after the migration of p to $X(p)$ has finished, meanwhile this resource must be reserved for p in both machines. The "transient constraints" are extensions of the capacity constraints for transient resources. Let $T \subseteq R$ be the set of transient resources, the transient constraints can be stated as

$$\sum_{p \in P | I_p = m \vee X(p) = m} D_{pr} \leq C_{mr}, \quad \forall\, m \in M, \forall\, r \in T. \tag{2}$$

Let Q be a partition of P into "services," such that $q \subseteq P$ for all $q \in Q$, $\bigcup_{q \in Q} q = P$, and $q' \cap q'' = \emptyset$ for all $q' \in Q$ and $q'' \in Q \backslash \{q'\}$. The "conflict constraints" enforce that two processes from the same service do not run on the same machine. These constraints are as follows:

$$X(p') \neq X(p''), \quad \forall\, q \in Q, \forall\, p' \in q, \forall\, p'' \in q \backslash \{p'\}. \tag{3}$$

Let L be a partition of M into "locations," such that $l \subseteq M$ for all $l \in L$, $\bigcup_{l \in L} l = M$, and $l' \cap l'' = \emptyset$ for all $l' \in L$ and $l'' \in L \backslash \{l'\}$. The "spread constraints" ensure that the processes of service $q \in Q$ are spread in at least $K_q \in \mathbb{N}$ distinct locations. Given $Y(q, l)$, such that $Y(q, l) = 1$ if there is at least one process of q in l on the reassignment X, and $Y(q, l) = 0$ otherwise, the spread constraints can be formulated as

$$\sum_{l \in L} Y(q, l) \geq K_q, \quad \forall\, q \in Q. \tag{4}$$

Some services "depend" on other services. Let N be a partition of M into "neighborhoods," such that $n \subseteq M$ for all $n \in N$, $\bigcup_{n \in N} n = M$, and $n' \cap n'' = \emptyset$ for all $n' \in N$ and $n'' \in N \backslash \{n'\}$. The "dependency constraints" ensure that if a service $q' \in Q$ depends on $q'' \in Q$ then each process of q' runs in the same neighborhood of a q'' process. Given the set E of tuples $(q', q'') \in Q^2$ such that the service q' depends on q'', the dependency constraints can be stated as

$$\exists p' \in q' | X(p') \in n \Rightarrow \exists p'' \in q'' | X(p'') \in n, \quad \forall\, (q', q'') \in E, \forall\, n \in N. \tag{5}$$

The MRP aims at improving the usage of the machine resources. The quality of the reassignment X is measured by five costs: "load cost," "balance cost," "process move cost," "machine move cost," and "service move cost." The total objective cost of MRP is a weighted sum of these individual costs. They are described as follows.

Let $S_{mr} \in \mathbb{N}$ be a target value for the maximum usage of a resource $r \in R$ for a machine $m \in M$, known as "safety capacity" of r for m, where $S_{mr} \leq C_{mr}$. The so-called "load cost" of $r \in R$ is defined as $LC(r) = \sum_{m \in M} \max(0, U(m, r) - S_{mr})$. The "total load cost" (TLC) is the sum of the load cost

for all resources. It is defined in the following equation, where $W_r^{LC} \in \mathbb{N}$ is the weight associated to $LC(r)$ in the total objective cost of MRP.

$$TLC = \sum_{r \in R} W_r^{LC} LC(r). \tag{6}$$

The "balance cost" measures the balance on the availability of some pairs of resources, in order to ensure that a machine has enough of both resources to run future upcoming processes. For example, it is not convenient that a machine has CPU resource available, but no RAM memory availability. Let $A(m, r) = C_{mr} - U(m, r)$ be the availability of a resource $r \in R$ for a machine $m \in M$. Given a set J of triples $\langle r', r'', t \rangle$, where $r' \in R$, $r'' \in R \backslash \{r'\}$, and $t \in \mathbb{N}$ is the target ratio for the availability of r' and r'', the "balance cost" for a triple $\langle r', r'', t \rangle \in J$ is defined as $BC(\langle r', r'', t \rangle) = \sum_{m \in M} \max(0, tA(m, r') - A(m, r''))$. The "total balance cost" (TBC) is the sum of the balance cost for all triples in J. It is given by the following equation where $W_j^{BC} \in \mathbb{N}$ is the weight associated to $BC(j)$ in the total objective cost of MRP.

$$TBC = \sum_{j \in J} W_j^{BC} BC(j). \tag{7}$$

Although moving processes from their initial machine can reduce the load and balance costs, it also implies additional costs of migration. The three move costs are described as follows. Each process $p \in P$ has its own individual "process move cost," denoted by $\mathcal{C}_p^{PMC} \in \mathbb{N}$, that models the cost of moving p from its initial machine. The "total process move cost" (TPMC) is the sum of the process move cost for all processes that changed machines. It is outlined in the following equation, where $W^{PMC} \in \mathbb{N}$ is the weight associated to the process move cost in the total objective cost of MRP.

$$TPMC = W^{PMC} \sum_{p \in P | I_p \neq X(p)} \mathcal{C}_p^{PMC}. \tag{8}$$

There is also a cost associated to the migration of any process from a machine $m' \in M$ to $m'' \in M \backslash \{m'\}$. It is called "machine move cost" and is denoted by $\mathcal{C}_{m'm''}^{MMC} \in \mathbb{N}$. The "total machine move cost" (TMMC) is the sum of the machine move cost for all processes that changed machines. It is shown in the following equation, where $W^{MMC} \in \mathbb{N}$ is the weight associated to the machine move cost in the total objective cost of MRP.

$$TMMC = W^{MMC} \sum_{p \in P | I_p \neq X(p)} \mathcal{C}_{I_p X(p)}^{MMC}. \tag{9}$$

Moving many processes of the same service is not desirable. The "service move number" of a service $q \in Q$, denoted by $SMN(q)$, is equal to the number of processes in q that have changed machines. The so-called service move cost (Equation 10) is defined as the maximum of $SMN(q)$ for all $q \in Q$, where $W^{SMC} \in \mathbb{N}$ is the weight associated to the service move cost in the total objective cost of MRP.

$$SMC = W^{SMC} \max_{q \in Q} SMN(q). \tag{10}$$

The MRP can now be formally defined as follows. Given a feasible initial process-to-machine assignment I, MRP aims at finding a reassignment X that does not violate constraints (1)–(5) and that minimizes the sum of (6)–(10). Next we discuss related works found in the literature.

MRP is closely related to the multiresource generalized assignment problem (MRGAP; Gavish and Pirkul, 1991). MRGAP consists in finding a minimum cost assignment of a set of tasks to a set of agents where the resource demands of each task are satisfied and the resource capacities of each agent are met. Gavish and Pirkul (1991) proposed heuristics and a branch-and-bound algorithm based on relaxations of the problem.

Another problem related to MRP is the bin packing problem with conflicts (BPPC; Jansen and Öhring, 1997). Given a set of items with sizes and a conflict graph, BPPC seeks a partition of the items into independent sets over the conflict graph, where the number of independent sets is minimized. Jansen and Öhring (1997) proposed approximation algorithms for BPPC, whereas Gendreau et al. (2004) proposed heuristics and developed valid lower bounds.

A related problem that arises in the context of data centers was studied in Chen et al. (2008). Given an initial assignment of processes to machines, the load rebalancing problem (LRP) consists in finding a process reassignment with a desired load balance that minimizes the process move cost. Chen et al. (2008) proposed a heuristic algorithm that makes use of local search procedures based on "shift" and "swap" neighborhoods. Computation experiments showed that this heuristic outperformed previous greedy and local search heuristics proposed for similar problems.

Gavranović et al. (2012) proposed a variable neighborhood search (VNS) heuristic for MRP. The local search is based on the "shift," "swap," "chain," and "BPR" neighborhoods. The "shift" neighborhood of a solution s is defined as the set of solutions that differ from s by the assignment of one process, while the the "swap" neighborhood of a solution s is defined as the set of solutions that differ from s by the machine swap of two processes. In the "chain" neighborhood, the set of neighbors of a solution s is obtained by shifting a specific number of processes in a given order, while in the "BPR" neighborhood, the set of neighbors of a solution s is obtained by moving some processes to a single machine and moving processes from this machine to another. The perturbation procedure used in this VNS heuristic applies the same local search to the current solution, but with a modified objective function in which the load cost weight (W_r^{LC}) of some resources is increased.

Mehta et al. (2012) proposed two large neighborhood search approaches for the MRP. The first approach consists in (a) selecting a subset of machines by solving a small combinatorial optimization problem, (b) creating a MRP subproblem, and (c) solving the subproblem by a constraint programming (CP) based algorithm. The second approach is similar to the first one, except that it solves the subproblem by a mixed integer programming (MIP) based algorithm. Computational experiments showed that the CP-based approach outperformed the MIP-based approach in all except two instances considered.

The most recent heuristic in the literature of MRP was proposed by Masson et al. (2013). MS-ILS-PP is a multistart heuristic for MRP based on iterated local search (ILS; Lourenço et al., 2003). The local search used explores the "shift" and "swap" neighborhoods, and the perturbation procedure used is based on two operators. The home relocate operator selects a given number of processes that are not assigned to their initial machines and moves them back to their initial machines, while the K-machine-swap operator randomly selects pairs of machines and for each pair of machines randomly selected pairs of processes are swapped.

In this paper, we propose a linear integer programming (IP) formulation for MRP in Section 2. As the state-of-the-art IP solvers could not solve the large size instances proposed in the Google ROADEF/EURO Challenge (2012), we propose four heuristics based on ILS in Section 3. Two of these heuristics are based on standard randomized perturbations, while the other two are based on the IP-based perturbations. The latter are also referred as "matheuristics" because they rely on a mathematical formulation. Our approach differ from the previous approches mainly by the IP formulation that is used only in the perturbation procedure, and by the restricted local search procedures that are efficient to identify and only evaluate the most promising neighbors in each neighborhood explored. Computational experiments that evaluate the performance of our heuristics and matheuristics are reported in Section 4, and concluding remarks are drawn in the last section.

2. Integer programming formulation

In this section, we propose an IP formulation for MRP. This formulation is used in the solution approach proposed in the next section. The reassignment X is modeled by variables x_{pm}, such that $x_{pm} = 1$ if process $p \in P$ is assigned to machine $m \in M$ on X, and $x_{pm} = 0$ otherwise. In order to formulate the spread constraints, we use auxiliary variables y_{ql}, such that $y_{ql} = 1$ if there is at least one process of service $q \in Q$ in location $l \in L$, and $y_{ql} = 0$ otherwise. We also need additional helping variables to formulate the objective function. First, the load cost of a resource $r \in R$ for a machine $m \in M$ is modeled by variables $z_{mr} \in \mathbb{N}$. Next, the balance cost of a triple $j \in J$ for a machine $m \in M$ is modeled by variables $w_{jm} \in \mathbb{N}$. Finally, the service move cost is held in variable $smc \in \mathbb{N}$. Our IP formulation of MRP is presented below.

$$\begin{aligned}\min \quad & \sum_{r\in R}\sum_{m\in M} W_r^{LC} z_{mr} \\ & + \sum_{j\in J}\sum_{m\in M} W_j^{BC} w_{jm} \\ & + W^{PMC}\left(\sum_{p\in P}(1 - x_{pI_p})\cdot \mathcal{C}_p^{PMC}\right) \\ & + W^{MMC}\left(\sum_{p\in P}\sum_{m\in M} x_{pm}\cdot \mathcal{C}_{I_p m}^{MMC}\right) \\ & + W^{SMC}\cdot smc \end{aligned} \tag{11}$$

subject to

$$\sum_{m\in M} x_{pm} = 1, \quad \forall\, p \in P \tag{12}$$

$$\sum_{p\in P} D_{pr} x_{pm} \le C_{mr}, \quad \forall\, m \in M, \forall\, r \in R \setminus T \tag{13}$$

$$\sum_{p\in P|I_p\neq m} D_{pr}x_{pm} + \sum_{p\in P|I_p=m} D_{pr} \leq C_{mr}, \quad \forall\, m \in M, \forall\, r \in T \tag{14}$$

$$\sum_{p\in q} x_{pm} \leq 1, \quad \forall\, m \in M, \forall\, q \in Q \tag{15}$$

$$\sum_{m\in n} x_{p'm} \leq \sum_{m\in n}\sum_{p''\in q''} x_{p''m}, \quad \forall\, n \in N, \forall\, (q', q'') \in E, \forall\, p' \in q' \tag{16}$$

$$K_q \leq \sum_{l\in L} y_{ql}, \quad \forall\, q \in Q \tag{17}$$

$$y_{ql} \leq \sum_{p\in q}\sum_{m\in l} x_{pm}, \quad \forall\, q \in Q, \forall\, l \in L \tag{18}$$

$$\sum_{p\in P} D_{pr} \cdot x_{pm} - S_{mr} \leq z_{mr}, \quad \forall\, m \in M, \forall\, r \in R \tag{19}$$

$$t \cdot A(m, r') - A(m, r'') \leq w_{jm}, \quad \forall\, j = \langle r', r'', t\rangle \in J \tag{20}$$

$$\sum_{p\in q}(1 - x_{pI_p}) \leq smc, \quad \forall\, q \in Q. \tag{21}$$

The objective function (11) is the sum of (a) the TLC, (b) the TBC, (c) the TPMC, (d) the TMMC, and (e) the service move cost, respectively. Equality (12) guarantees that each process is assigned to a single machine. The capacity constraints and transient constraints are formulated in Equations (13) and (14), respectively. Equations (15) and (16) enforce the conflict constraints and the dependency constraints, respectively. Inequalities (17) and (18) together assure the spread constraints. Equations (19)–(21) define, respectively, the value of variables z_{mr}, w_{jm}, and smc, where $A(m, r) = C_{mr} - \sum_{p\in P} D_{pr}x_{pm}$.

3. Solution approach

We propose four heuristics based on ILS (Lourenço et al., 2003). ILS performs a search in the space of local optima. It starts from a local optimum s and, at each iteration, performs a perturbation in this solution followed by a local search in the perturbed solution, in order to produce a new local optimum s'. If s' meets some acceptance criterion, the procedure is repeated from s', otherwise the heuristic continues from s. The ILS-based heuristics were successfully used for solving assignment and scheduling problems, such as the quadratic assignment problem (Hussin and Stützle, 2009;

Stützle, 2006), the referee assignment problem (Duarte et al., 2007), and the car sequencing problem (Aloise et al., 2008; Ribeiro et al., 2008). The local searches and perturbation procedures used in the ILS heuristics proposed in this paper are described below.

The "shift" local search explores the complete "shift" neighborhood. For each process $p \in P$, the "shift" local search evaluates, at each iteration, the overall cost of migrating p to every machine in M. Then it moves p to the machine that results in the minimum overall cost, if the resulting solution cost is lower than the current one. The size of the "shift" neighborhood is $O(|P| \cdot |M|)$, and the worst-case complexity of evaluating the cost and checking the feasibility of one "shift" neighbor is $O(|R| + |Q|)$.

The "swap" local search explores the complete "swap" neighborhood. For each pair of processes $p', p'' \in P$, the "swap" local search evaluates, at each iteration, the overall cost of migrating p' to m'' and p'' to m', where m' and m'' are, respectively, the machines assigned to p' and p'', and performs the first swap that results in a solution with lower cost than the current one. The size of the "swap" neighborhood is $O\left(|P|^2\right)$, and the worst-case complexity of evaluating the cost and checking the feasibility of one *swap* neighbor is $O(|R| + |Q|)$.

For the sake of efficiency in solving the large instances with up to 50,000 processes and 5000 machines proposed in the Google ROADEF/EURO Challenge (2012), we propose more efficient local search procedures that are restricted versions of the previous local searches. Instead of exploring all neighbors at each iteration of the local search, the so-called "restricted local searches" select a subset of neighbors of the current solution. Therefore, they are expected to reach a local optimum (for the restricted neighborhood) faster than the standard local searches.

The "shift machine load sort" local search is an efficient implementation of the "shift" local search. Instead of evaluating all "shift" neighbors arbitrarily, this local search evaluates only a subset of them. Besides, the neighbors that are more likely to have lower costs are evaluated earlier than the others, resulting in a faster convergence to a local optimum. In this procedure, the machines in M are sorted by nonincreasing order of $TLC(m) = \sum_{r \in R} W_r^{LC} LC_m(r)$, where $LC_m(r) = \max(0, U(m, r) - S_{mr})$ for all $m \in M$, that is, the machines that are more saturated come first in the resulting machine permutation. The migration of a process from a machine with higher values of TLC to a machine with smaller values of TLC is more likely to result in a solution with lower load cost than the migration of a random process to a random machine. Therefore, we only evaluate the migration of processes from machines with higher values of TLC to machines with smaller values of TLC. Moreover, for each machine, we sort their respective processes by a nonincreasing order of $TR(p) = \sum_{r \in R} W_r^{LC} D_{pr}$, that is, the processes that demand more resources come first in the respective process permutation of each machine. Processes that demand more resources are harder to move than processes that demand less resources. Therefore, we first evaluate the cost of migrating the processes from the first half of the process permutation of each machine. When a local optimum from this restricted neighborhood is found, the other processes are considered. As the latter are expected to be easier to migrate, it is expected that further improvements can be obtained. The worst-case complexity of evaluating the cost and checking the feasibility of one "shift machine load sort" neighbor is $O(|R| + |Q|)$.

The "swap in service" neighborhood of a solution s is defined as the set of solutions that differ from s by the machine swap of two processes in the same service $q \in Q$. The "swap in service" local search explores the "swap in service" neighborhood. Given a service $q \in Q$, for each pair of processes $p', p'' \in q$, the "swap in service" local search evaluates, at each iteration, the overall cost

of swapping the processes p' and p'', and performs the first swap that results in a solution whose overall cost is lower than the cost of the current solution. Since two processes from the same service do not violate dependency, spread, and conflict constraints, the computational cost of checking the solution feasibility can be spared. The number of "swap in service" neighbors of a solution is $O\left(|q|^2\right)$, and the worst-case complexity of evaluating the cost and checking the feasibility of one "swap in service" neighbor is $O\left(|R|\right)$.

The "restricted swap" local search consists in a standard "swap" local search on a restricted subset of the "swap" neighborhood. At the beginning of this local search, a set $P' \subset P$ is selected. In our case, we select only the processes that changed machines in the perturbed solution of the ILS heuristics. This choice is motivated by the fact that the processes in P' are more likely to result in solutions with lower costs when moved than the other processes. For each pair of processes $p' \in P'$ and $p'' \in P \setminus P'$, the "restricted swap" local search evaluates, at each iteration, the overall cost of swapping the processes p' and p'', and performs the first swap that results in a solution whose overall cost is lower than the cost of the current solution. The worst-case complexity of evaluating the cost and checking the feasibility of one "restricted swap" neighbor is $O\left(|R| + |Q|\right)$.

We propose four perturbation procedures based on the neighborhoods and the mathematical formulation defined above. Each one is used in a different ILS heuristic. The four perturbation procedures are described below.

The "k-shift" perturbation consists in selecting a "k-shift" neighbor of a solution at random. For the sake of efficiency in checking the feasibility of the resulting solution, we consider only the subset of solutions in this neighborhood that can be obtained by k-successive "shift" movements. The worst-case complexity of generating a perturbed solution (from the current one) is $O(k(|R| + |Q|))$.

The so-called *IP* perturbation is a procedure based on the IP formulation presented in Section 2. This perturbation selects a subset $M' \subset M$ of machines at random and builds a restricted MRP similar to the original MRP. Let $P' \subset P$ be the set of processes assigned to the machines in M', the restricted MRP consists in finding the optimal reassignment for the processes in P' to the machines in M'. This restricted problem is formulated by fixing $x_{pm} = 1$ for every process $p \in P \setminus P'$, where m is the machine assigned to p in s and fixing $x_{p'm} = 0$ for all $p' \in P'$ and $m \in M \setminus M'$. The resulting IP problem is solved by a branch-and-bound algorithm implemented by the state-of-the-art IP solver CPLEX. For the sake of efficiency, we stop the branch-and-bound algorithm in case it takes more than $\gamma = 5$ seconds for solving the restricted problem and use the best integer solution found.

We also propose more efficient perturbation procedures that are more suitable to solve the large instances proposed in the Google ROADEF/EURO Challenge (2012). The "k-swap in service" perturbation selects a "k-swap in service" neighbor of a solution at random. For the sake of efficiency, we consider only solutions in this neighborhood that can be obtained by k-successive "swap in service" movements. As two processes from the same service do not violate dependency, spread, and conflict constraints, the computational cost of checking the solution feasibility at each successive movement can be spared. The worst-case complexity of generating a perturbed solution in the "k-swap in service" perturbation is $O(k|R|)$.

The so-called "restricted IP perturbation" is similar to the IP perturbation described above. However, in this case, the machines in the subset M' are randomly selected from the same neighborhood $n \in N$, which is also randomly selected. This allows us to remove the dependency constraints from the formulation of Section 2, which leads to an easier to solve IP formulation.

```
begin ILS(s^i)
1.  s, s* ← LocalSearch(swap, s^i);
2.  while stopping condition is not met do
3.      while stagnation condition is not met do
4.          s' ← Perturbation(s);
5.          s' ← LocalSearch(shift, s');
6.          if AcceptanceCriterion(s, s') then s ← s';
7.          if s' is better than s* then s* ← s';
8.      end-while;
9.      s ← LocalSearch(swap, s);
10.     if s is better than s* then s* ← s;
11. end-while;
12. return s*;
end
```

Fig. 1. Pseudocode of k-ILS and IP-ILS. In the k-ILS heuristic, the perturbation procedure (line 4) is the "k-shift" perturbation, while in the IP-ILS heuristic, the IP perturbation is applied.

We have two types of ILS heuristics: the so-called "unrestricted ILS heuristics" and the "restricted ILS heuristics." The former uses the local searches based on the "shift" and "swap" neighborhoods, while the latter uses the local searches based on neighborhoods "shift machine load sort," "swap in service," and "restricted swap."

The unrestricted ILS heuristics are called k-ILS and IP-ILS. The former uses the "k-shift" perturbation, while the latter uses the IP perturbation. The pseudocode of these two heuristics is described in Fig. 1. The algorithm starts from the initial solution s^i that consists of the initial assignment of processes to machines. Then, the "swap" local search procedure is applied to s^i in line 1, in order to produce the first local optimum s. The loop in lines 2–11 is performed until a stopping condition is met. A new local optimum s' is obtained by performing a perturbation procedure to s in line 4, followed by the "shift" local search procedure to the perturbed solution in line 5. In the k-ILS heuristic, the "k-shift" perturbation is applied, with $k = 0.05 \cdot |P|$. In the IP-ILS, heuristic the IP perturbation is applied to $\delta = 7$ machines selected at random. The current local optimum s is replaced by s' accordingly to a defined acceptance criterion in line 6. s' is accepted if its cost is not larger than $\alpha = 0.1 \cdot F(s)$, where $F(s)$ is the cost of s. The best-known solution is updated in line 7. The loop in lines 3–8 is repeated until a stagnation condition is met, which in this case is $\beta = 10$ iterations without improving the best-known solution. Once the stagnation condition is met, a "swap" local search procedure is applied to the current solution in line 9. This procedure consumes considerably more CPU time than the "shift" local search procedure, but the cost of the "swap" local optima are usually smaller than or equal to that of the "shift" local optima. The best solution found by this heuristic is returned in line 12.

The restricted ILS heuristics are called k-RILS and IP-RILS. The former uses the "k-swap in service" perturbation, while the latter uses the restricted IP perturbation. The pseudocode of these two heuristics is described in Fig. 2. They start from the initial solution s^i that consists of the initial assignment of processes to machines. Then, the "shift machine load sort" local search is applied to s^i in line 1, in order to produce the first local optimum s. The loop in lines 2–8 is performed until a

```
begin RILS(s^i)
1. s, s* ← LocalSearch(shift machine load sort, s^i);
2. while stopping condition is not met do
3.     s' ← Perturbation(s);
4.     s' ← VND(shift machine load sort, swap in service, s');
5.     s' ← LocalSearch(restricted swap, s');
6.     if AcceptanceCriterion(s, s') then s ← s';
7.     if s' is better than s* then s* ← s';
8. end-while;
9. return s*;
end
```

Fig. 2. Pseudocode of k-RILS and IP-RILS. In the k-RILS heuristic, the perturbation procedure (line 3) is the "k-swap in service" perturbation, while in the IP-RILS heuristic the restricted IP perturbation is applied.

stopping condition is met. A new solution s' is obtained by performing a perturbation procedure to s in line 3. In the k-RILS heuristic, the "k-swap in service" perturbation is applied, with $k = 0.10|P|$. In the IP-RILS heuristic the restricted IP perturbation is applied to $\delta = 5$ machines selected at random from the same randomly selected neighborhood $n \in N$. A variable neighborhood descent (VND; Hansen and Mladenović, 2001a,b) is applied to s' in line 4, replacing s' by a local optimum of both "shift machine load sort" and the "swap in service" neighborhoods. A "restricted swap" local search is applied to the resulting solution in line 5. This local search was not included in the VND procedure because it consumes considerably more CPU time than the local searches "shift machine load sort" and "swap in service." Besides, we observed that the latter are not likely to improve a "restricted swap" local optimum. The current local optimum s is replaced by s' accordingly to a defined acceptance criterion in line 6. s' is accepted if its cost is not larger than $\alpha = 0.1F(s)$, where $F(s)$ is the cost of s. The best-known solution s^* is updated in line 7 and returned in line 9.

4. Computational experiments

Computational experiments were performed on a 64 bits E8400 Intel® Core™ 2 Duo with 3.0 GHz of clock and 4 GB of RAM memory, running Linux operating system. k-ILS, IP-ILS, k-RILS, and IP-RILS were implemented in C++, and compiled with GCC 4.4.3. Our implementation of IP-ILS and IP-RILS makes use of ILOG CPLEX version 12.3 with default parameter settings.

Computational experiments were performed on the instances proposed in the Google ROADEF/EURO Challenge (2012). Two sets of instances, named B and X, were used for evaluating the teams in the competition. Table 1 shows the details of both sets of instances. The first column shows the instance name. The number of processes, machines, and resources are presented in columns 2–4, respectively. The next three columns display, respectively, the number of services, locations, and neighborhoods. The number of tuples in J is shown in the last column.

In the first experiment, the stopping criterion of the four heuristics was set to 300 seconds of CPU time, as suggested by the Google ROADEF/EURO Challenge (2012). The heuristics were run 25 times for each instance, varying the seed of the random number generator. The results for the

Table 1
Description of the sets of instances used in the computational experiments

Name	$\|P\|$	$\|M\|$	$\|R\|$	$\|Q\|$	$\|L\|$	$\|N\|$	$\|J\|$
B_1	5000	100	12	2512	10	5	0
B_2	5000	100	12	2462	10	5	1
B_3	20,000	100	6	15,025	10	5	0
B_4	20,000	500	6	1732	10	5	1
B_5	40,000	100	6	35,082	10	5	0
B_6	40,000	200	6	35,082	50	5	1
B_7	40,000	4000	6	15,050	50	5	1
B_8	50,000	100	3	45,030	10	5	0
B_9	50,000	1000	3	4609	100	5	1
B_10	50,000	5000	3	4896	100	5	1
X_1	5000	100	12	2529	10	5	0
X_2	5000	100	12	2484	10	5	1
X_3	20,000	100	6	14,928	10	5	0
X_4	20,000	500	6	1190	10	5	1
X_5	40,000	100	6	34,872	10	5	0
X_6	40,000	200	6	14,504	50	5	1
X_7	40,000	4000	6	15,273	50	5	1
X_8	50,000	100	3	44,950	10	5	0
X_9	50,000	1000	3	4871	100	5	1
X_10	50,000	5000	3	4615	100	5	1

unrestricted ILS heuristics are displayed in Table 2, while the results for the restricted ILS heuristics are shown in Table 3. In both tables, the first two columns present, respectively, the instance name and the cost of the best solution known for each instance, that is, the best solution found in any run of all algorithms submitted to the Google ROADEF/EURO Challenge (2012). Columns 3–6 of Table 2 (resp. Table 3) show, respectively, the cost of the best solution found, the average cost of the solutions provided by k-ILS (resp. k-RILS), the coefficient of variation ($CV = \sigma/F^H$), and the relative gap ($(F^H - F^{best})/F^{best}$) of the average cost of the solutions provided by the heuristic (F^H) and the cost of the best solution known (F^{best}), where σ stands for the standard deviation. The same results are displayed for IP-ILS (resp. IP-RILS) in columns 7–10. It can be observed that the restricted ILS heuristics outperformed the unrestricted ILS heuristics, since the average relative gaps and CV of k-RILS and IP-RILS were smaller than those of k-ILS and IP-ILS. One can also observe that the ILS heuristics with IP-based perturbation outperformed the ILS heuristics with randomized perturbation, since IP-ILS obtained better results than k-ILS in 15 (out of the 20) instances, and IP-RILS obtained better results than k-RILS in all but instances B_3 and X_10. We note that the large average gaps on instances X_3, X_5, and X_8 are due to the very large values of the "load cost weight" (W_r^{LC}) relatively to the other cost weights. Therefore, little differences in the assignment of two solutions can result in a huge variation of the solution cost. This was also observed in other heuristics for MRP in the literature (Masson et al., 2013).

In the second experiment, the restricted ILS heuristics, k-RILS, and IP-RILS were set to stop whenever a solution with cost smaller than or equal to a given target cost was found. The target

Table 2
Comparison of the average cost of the reassignment provided by the algorithms *k*-ILS and IP-ILS on sets B and X

Instance name	Best solution among all teams	*k*-ILS				IP-ILS			
		Best	average	CV	Gap (%)	Best	average	CV	Gap (%)
B_1	3,339,186,879	3,909,987,252	4,058,255,841.400	0.017	21.534	3,986,173,310	4,050,347,667.640	0.008	21.297
B_2	1,015,553,800	1,017,058,482	1,017,058,482.000	0.000	0.148	1,016,072,693	1,016,694,358.840	0.000	0.112
B_3	156,835,787	328,150,797	337,534,416.320	0.015	115.215	283,579,861	300,012,710.520	0.026	91.291
B_4	4,677,823,040	4,678,159,738	4,678,164,613.160	0.000	0.007	4,677,952,033	4,677,966,364.880	0.000	0.003
B_5	923,092,380	1,171,048,828	1,254,789,673.000	0.034	35.933	1,072,299,350	1,134,098,731.480	0.039	22.859
B_6	9,525,857,752	9,525,881,530	9,526,355,357.080	0.000	0.005	9,525,876,735	9,525,879,613.960	0.000	0.000
B_7	14,835,149,752	15,099,009,268	15,100,705,338.400	0.000	1.790	14,835,395,282	14,835,611,651.960	0.000	0.003
B_8	1,214,458,817	1,214,659,766	1,223,697,550.680	0.020	0.761	1,214,574,232	1,214,630,761.200	0.000	0.014
B_9	15,885,486,698	15,886,814,569	15,886,815,555.040	0.000	0.008	15,885,956,462	15,885,961,685.040	0.000	0.003
B_10	18,048,515,118	18,136,885,141	18,137,735,433.960	0.000	0.494	18,048,805,847	18,048,847,782.000	0.000	0.002
X_1	3,100,852,728	3,820,203,848	3,947,974,380.400	0.015	27.319	4,034,765,304	4,034,765,304.000	0.000	30.118
X_2	1,002,502,119	1,022,553,344	1,022,553,344.000	0.000	2.000	1,011,856,297	1,011,863,849.040	0.000	0.934
X_3	211,656	66,073,706	79,776,357.680	0.038	37,591.517	2,152,828	2,266,117.160	0.032	970.660
X_4	4,721,629,497	4,721,927,689	4,721,932,072.120	0.000	0.006	4,721,780,435	4,721,908,058.720	0.000	0.006
X_5	93,823	13,272,266	134,556,993.680	0.494	143,315.787	5,696,416	14,168,874.680	2.858	15,001.707
X_6	9,546,941,232	9,546,953,984	9,547,306,024.600	0.000	0.004	9,546,958,474	9,547,432,292.320	0.000	0.005
X_7	14,253,273,178	14,494,185,247	14,494,434,062.000	0.000	1.692	14,495,020,777	14,495,063,347.440	0.000	1.696
X_8	42,674	566,778	13,822,146.080	2.179	32,290.088	163,332	74,873,827.080	0.302	175,355.376
X_9	16,125,612,590	16,126,950,555	16,126,951,216.320	0.000	0.008	16,126,951,545	16,126,952,317.200	0.000	0.008
X_10	17,816,514,161	17,936,180,307	17,936,263,986.040	0.000	0.672	17,936,676,184	17,936,799,291.160	0.000	0.675
Average				0.141	10,670.250			0.163	9,574.839

Table 3
Comparison of the average cost of the reassignment provided by the algorithms *k*-RILS and IP-RILS on sets B and X

Instance name	Best solution among all teams	*k*-RILS Best	*k*-RILS average	*k*-RILS CV	*k*-RILS Gap (%)	IP-RILS Best	IP-RILS average	IP-RILS CV	IP-RILS Gap (%)
B_1	3,339,186,879	3,609,450,429	3,701,479,764.840	0.014	10.850	3,511,150,815	3,607,699,652.200	0.009	8.041
B_2	1,015,553,800	1,033,694,006	1,037,543,009.280	0.002	2.165	1,017,134,891	1,020,525,550.200	0.002	0.490
B_3	156,835,787	163,611,167	168,157,870.240	0.030	7.219	161,557,602	181,080,108.080	0.082	15.458
B_4	4,677,823,040	4,678,225,748	4,678,280,005.960	0.000	0.010	4,677,999,380	4,678,007,686.840	0.000	0.004
B_5	923,092,380	933,583,379	936,779,779.320	0.002	1.483	923,732,659	925,603,724.960	0.002	0.272
B_6	9,525,857,752	9,525,987,366	9,526,007,155.080	0.000	0.002	9,525,937,918	9,525,944,361.560	0.000	0.001
B_7	14,835,149,752	14,835,734,054	14,839,155,139.920	0.000	0.027	14,835,597,627	14,836,339,894.240	0.000	0.008
B_8	1,214,458,817	1,217,533,874	1,227,743,620.560	0.004	1.094	1,214,900,909	1,222,143,151.200	0.007	0.633
B_9	15,885,486,698	15,885,787,050	15,885,814,581.400	0.000	0.002	15,885,632,605	15,885,650,287.120	0.000	0.001
B_10	18,048,515,118	18,051,150,431	18,063,431,912.400	0.001	0.083	18,052,239,907	18,061,426,028.400	0.000	0.072
X_1	3,100,852,728	3,378,498,525	3,432,204,088.600	0.007	10.686	3,341,920,446	3,449,508,630.360	0.013	11.244
X_2	1,002,502,119	1,022,260,881	1,028,859,183.360	0.003	2.629	1,008,340,365	1,018,358,697.000	0.006	1.582
X_3	211,656	2,586,119	4,458,937.120	0.250	2,006.691	1,359,493	3,107,399.400	0.284	1,368.137
X_4	4,721,629,497	4,722,159,549	4,722,159,549.000	0.000	0.011	4,721,833,040	4,721,845,449.920	0.000	0.005
X_5	93,823	825,687	905,908.200	0.042	865.550	385,150	424,661.440	0.070	352.620
X_6	9,546,941,232	9,547,073,463	9,547,073,463.000	0.000	0.001	9,547,002,140	9,547,008,878.720	0.000	0.001
X_7	14,253,273,178	14,254,171,473	14,255,576,238.240	0.000	0.016	14,253,835,332	14,254,276,126.480	0.000	0.007
X_8	42,674	336,662	336,662.000	0.000	688.916	96,936	111,407.240	0.074	161.066
X_9	16,125,612,590	16,125,991,006	16,126,021,528.600	0.000	0.003	16,125,780,091	16,125,794,190.640	0.000	0.001
X_10	17,816,514,161	17,818,606,909	17,821,836,504.200	0.000	0.030	17,819,116,915	17,821,801,412.880	0.000	0.030
Average				0.018	179.873			0.027	95.984

cost for each instance was set to the average solution cost obtained by 10 independent 300-second runs of *k*-RILS. The heuristics were run 100 times for each instance, varying the random number generator seed. Then, we draw the time-to-target value plots (TTT plots, for short; Aiex et al., 2006) for the two heuristics. TTT plot is a technique for comparing the performance of different stochastic algorithms for a given optimization problem. Given an instance and a target value for the cost of a solution for this instance, TTT plots display the empirical probability of an algorithm finding a solution whose cost is as good as the target within a given running time.

One can see from Fig. 3 that for some instances such as B_5 the probability of IP-RILS and *k*-RILS finding a solution as good as the target is almost the same. It can be seen from Fig. 4 (TTT plot for instance B_2) and Fig. 5 (TTT plot for instance B_8) that for some instances, such as B_2 and B_8, IP-RILS is more reliable than *k*-RILS, since the difference between the longest and the fastest runs of IP-RILS is smaller than that of *k*-RILS. Besides, it can also be seen that IP-RILS found solutions as good as the target with probability larger than 70% faster than *k*-RILS. Figures 6–8 show the TTT plot for instances X_3, X_7, and X_9, respectively. One can see that for these instances, the time-to-target probability of IP-RILS is clearly higher than that of *k*-RILS. Figures 9 and 10 display the TTT plots for instances X_1 and B_3, respectively. The latter are the only instances where *k*-RILS found better average results than IP-RILS (see Table 3). One can see that the time-to-target probability of *k*-RILS is smaller than that of IP-RILS for these instances. However, it can also be observed that the time-to-target of both heuristics with probability higher than 90% is equivalent.

The previous experiments showed that IP-RILS is the best among the four heuristics proposed in this paper. In the third experiment, we compare the results of IP-RILS with those of MS-ILS-PP

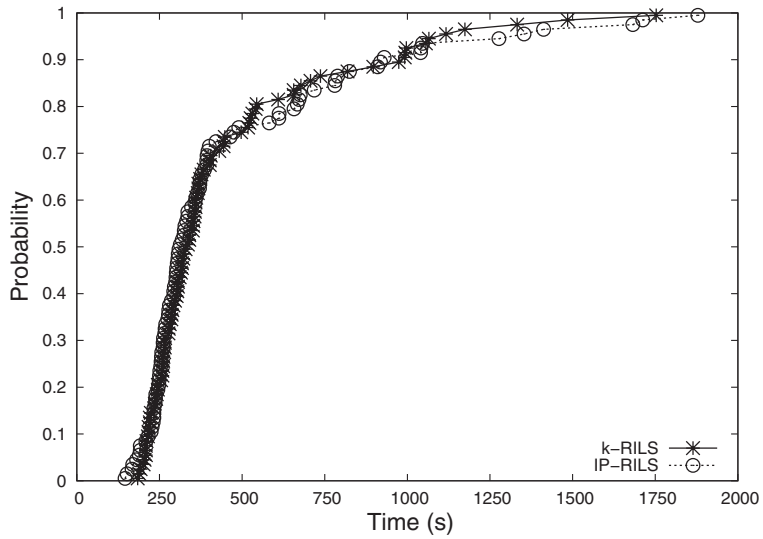

Fig. 3. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance B_5.

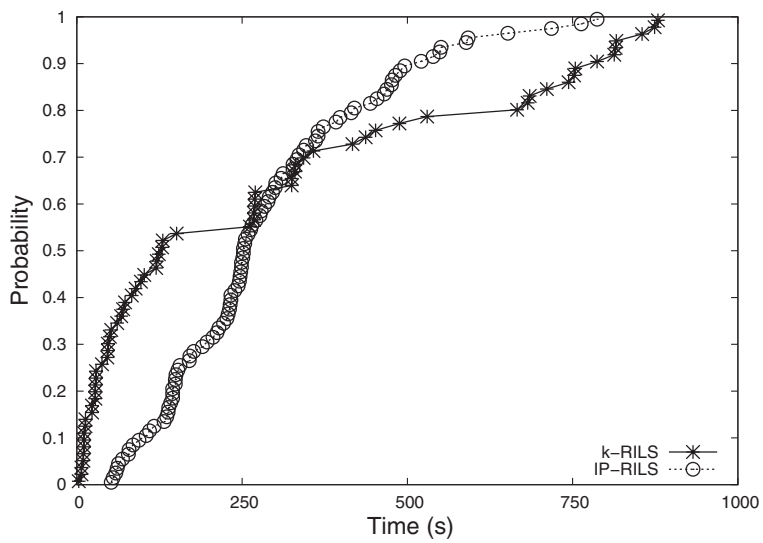

Fig. 4. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance B_2.

(Masson et al., 2013). The latter is the most recent heuristic in the literature of MRP. Besides, it is the only work in the literature that reports results for both sets B and X of instances. The first column of Table 4 gives the instance name. The cost of the initial solution and the cost of the best solution known for each instance is displayed in columns 2 and 3, respectively. Columns 4 to 6 show, respectively, the best cost, the average cost of the solutions, and the relative gap $((F^H - F^{best})/F^{best})$ provided by MS-ILS-PP (F^H) and the cost of the best solution known (F^{best}). Columns 7 to 9 show the same data for IP-RILS, respectively. It can be observed that the average relative gap of IP-RILS

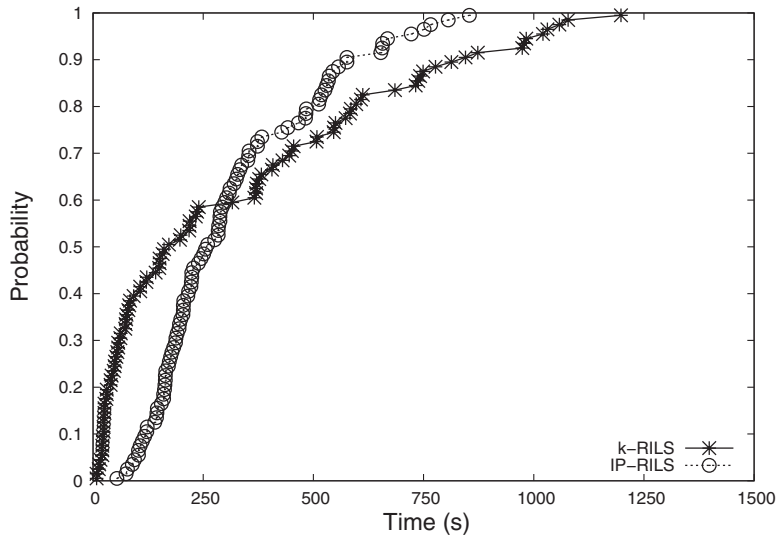

Fig. 5. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance B_8.

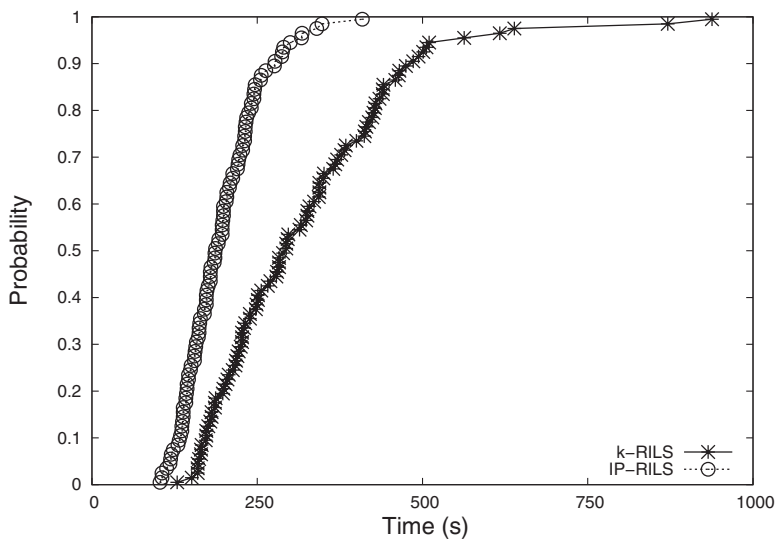

Fig. 6. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance X_3.

was smaller than that of MS-ILS-PP in 11 instances, while the average relative gap of MS-ILS-PP was smaller than that of IP-RILS in 5 instances. The average gap of MS-ILS-PP was 142%, while that of IP-RILS was 96%. These large gaps are due to the bad performance of both heuristics on instances X_3, X_5, and X_8. This is due to the fact that (i) the load cost weight (W_r^{LC}) is very large relatively to the other cost weights in the objective function, and that (ii) the heuristics failed to find solutions with smaller values of load cost than the best algorithms that participated in the Google ROADEF/EURO Challenge (2012) for these three instances on average. We also point out that the

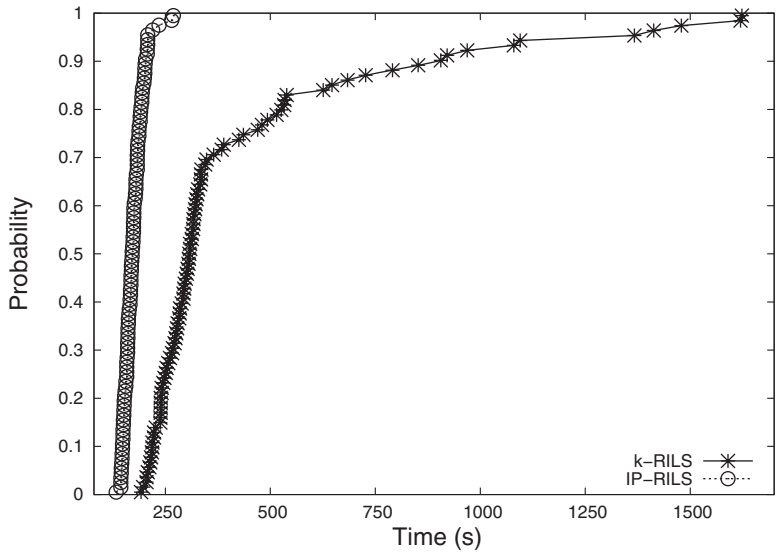

Fig. 7. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance X_7.

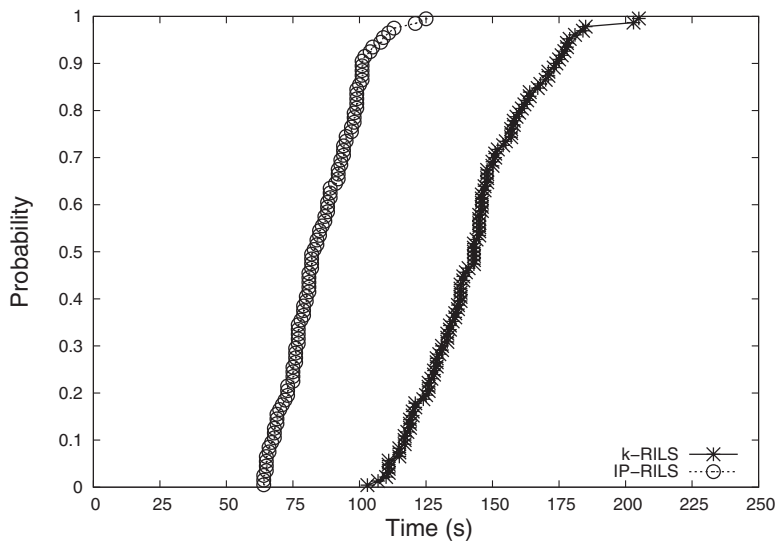

Fig. 8. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance X_9.

values reported in Column 3 are the best solution found in any run of all algorithms submitted to the Google ROADEF/EURO Challenge (2012). There is no guarantee that one of these algorithms find solutions as good as these on average. The average gaps of MS-ILS-PP and IP-RILS for all but these three instances were both close to 2%.

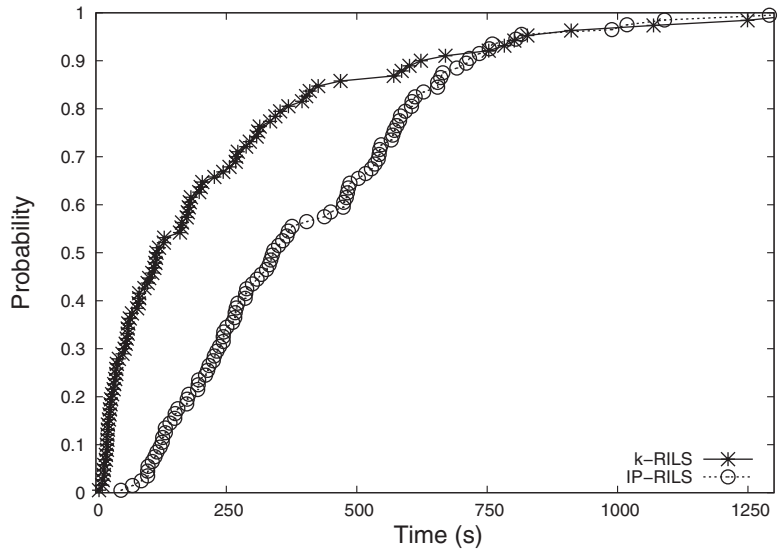

Fig. 9. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance X_1.

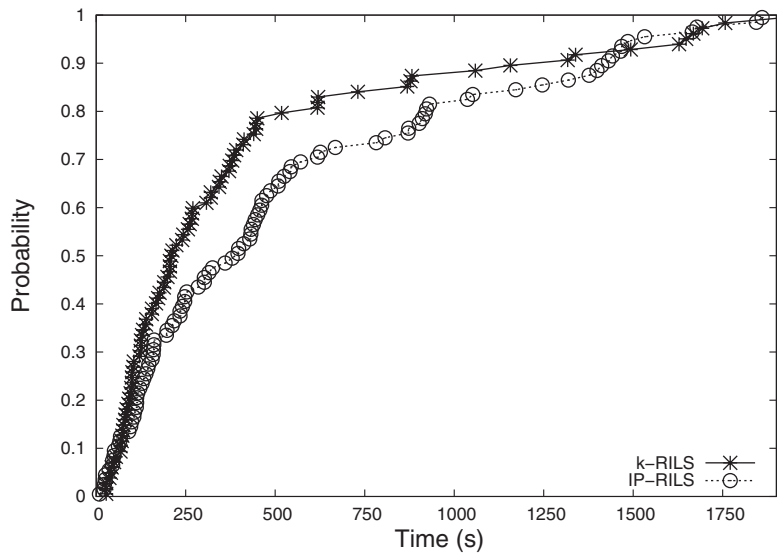

Fig. 10. The TTT plot for the algorithms *k*-RILS and IP-RILS on instance B_3.

5. Concluding remarks

In this paper, we proposed a linear IP formulation and four ILS heuristics for the MRP proposed in the Google ROADEF/EURO Challenge (2012). Although the CPLEX's branch-and-bound algorithm implemented with the proposed IP formulation could not solve the large real-life size instances with up to 50,000 processes and 5000 machines, it was able to solve restricted versions of the problem that allowed us to develop IP-based perturbation procedures. We also proposed

Table 4
Comparison of the average cost of the reassignment provided by the algorithm IP-RILS and the approach of Masson et al. (2013) on sets B and X

			MS-ILS-PP (Masson et al., 2013)			IP-RILS		
Instance name	Initial solution	Best solution among all teams	Best	Average	Gap (%)	Best	Average	Gap (%)
B_1	7,644,173,180	3,339,186,879	3,516,215,073	3,643,207,680	9.105	3,511,150,815	3,607,699,652.200	8.041
B_2	5,181,493,830	1,015,553,800	1,027,393,159	1,034,641,974	1.880	1,017,134,891	1,020,525,550.200	0.490
B_3	6,336,834,660	156,835,787	158,027,548	165,037,128	5.229	161,557,602	181,080,108.080	15.458
B_4	9,209,576,380	4,677,823,040	4,677,940,074	4,677,962,023	0.003	4,677,999,380	4,678,007,686.840	0.004
B_5	12,426,813,010	923,092,380	923,857,499	926,221,613	0.339	923,732,659	925,603,724.960	0.272
B_6	12,749,861,240	9,525,857,752	9,525,913,044	9,525,923,099	0.001	9,525,937,918	9,525,944,361.560	0.001
B_7	37,946,901,700	14,835,149,752	15,244,960,848	15,372,961,904	3.625	14,835,597,627	14,836,339,894.240	0.008
B_8	14,068,207,250	1,214,458,817	1,214,930,327	1,220,521,241	0.499	1,214,900,909	1,222,143,151.200	0.633
B_9	23,234,641,520	15,885,486,698	15,885,617,841	15,885,635,887	0.001	15,885,632,605	15,885,650,287.120	0.001
B_10	42,220,868,760	18,048,515,118	18,093,202,104	18,155,878,491	0.595	18,052,239,907	18,061,426,028.400	0.072
X_1	7,422,426,760	3,100,852,728	3,209,874,890	3,348,966,927	8.001	3,341,920,446	3,449,508,630.360	11.244
X_2	5,103,634,830	1,002,502,119	1,018,646,825	1,026,504,981	2.394	1,008,340,365	1,018,358,697.000	1.582
X_3	6,119,933,380	211,656	1,965,401	3,151,834	1,389.130	1,359,493	3,107,399.400	1,368.137
X_4	9,207,188,610	4,721,629,497	4,721,786,173	4,721,814,589	0.004	4,721,833,040	4,721,845,449.920	0.005
X_5	12,369,526,590	93,823	615,277	696,174	642.008	385,150	424,661.440	352.620
X_6	12,753,566,360	9,546,941,232	9,546,992,887	9,547,005,000	0.001	9,547,002,140	9,547,008,878.720	0.001
X_7	37,763,791,230	14,253,273,178	14,701,830,252	14,893,088,510	4.489	14,253,835,332	14,254,276,126.480	0.007
X_8	11,611,565,600	42,674	309,080	369,803	766.577	96,936	111,407.240	161.066
X_9	23,146,106,380	16,125,612,590	16,125,753,242	16,125,775,950	0.001	16,125,780,091	16,125,794,190.640	0.001
X_10	42,201,640,770	17,816,514,161	17,867,789,754	17,928,266,694	0.627	17,819,116,915	17,821,801,412.880	0.030
Average					141.725			95.984

efficient restricted versions of the classic perturbation and local search procedures based on the classic "shift" and "swap" neighborhoods.

Computational experiments showed that the ILS heuristic based on the restricted IP perturbation and the restricted local searches (IP-RILS) outperformed the other ILS heuristics based on the standard randomized perturbations and standard local search procedures. Besides, computational experiments showed that IP-RILS is competitive with other heuristics in the literature of MRPs. IP-RILS got the 14th prize among the 48 teams classified for the final phase of the Google ROADEF/EURO Challenge (2012).

The IP-based perturbation and restricted local search procedures proposed in this paper could also be used to solve large instances of related problems, such as the MRGAP (Gavish and Pirkul, 1991), the BPPC (Jansen and Öhring, 1997), the multicapacity bin packing problem (Masson et al., 2013), and load rebalancing problem (Chen et al., 2008). The results of our heuristics for MRP could be improved by smaller and tighter mathematical formulations, which would allow the heuristics to solve larger subproblems at each iteration. Also, restricted versions of the "chain" and "BPR" neighborhoods (Gavranovi et al., 2012) could be developed and added to our heuristics in order to improve their performance.

Acknowledgments

This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), the Foundation for Support of Research of the State of Minas Gerais,

Brazil (FAPEMIG), and Coordination for the Improvement of Higher Education Personnel, Brazil (CAPES).

References

Aiex, R., Resende, M., Ribeiro, C., 2006. TTT plots: a perl program to create time-to-target plots. *Optimization Letters* 1, 10–1007.

Aloise, D., Ribeiro, C., Noronha, T., Rocha, C., Urrutia, S., 2008. A hybrid heuristic for a multi-objective real-life car sequencing problem with painting and assembly line constraints. *European Journal of Operational Research* 191, 981–992.

Chen, L., Wang, C., Lau, F., 2008. Process reassignment with reduced migration cost in grid load rebalancing. *22nd IEEE International Symposium on Parallel and Distributed Processing*, Miami, FL, pp. 1–13.

Duarte, A., Ribeiro, C., Urrutia, S., 2007. A hybrid ILS heuristic to the referee assignment problem with an embedded MIP strategy. *Proceedings of the 4th International Conference on Hybrid Metaheuristics*, Springer-Verlag, Berlin, Heidelberg, pp. 82–95.

Gavish, B., Pirkul, H., 1991. Algorithms for the multi-resource generalized assignment problem, *Management Science* 37, 6, 695–713.

Gavranović, H., Buljubašić, M., Demirović, E., 2012. Variable neighborhood search for Google machine reassignment problem. *Electronic Notes in Discrete Mathematics* 39, 209–216.

Gendreau, M., Laporte, G., Semet, F., 2004. Heuristics and lower bounds for the bin packing problem with conflicts. *Computers & Operations Research* 31, 3, 347–358.

Google ROADEF/EURO Challenge, 2012. Machine reassignment. Available at http://challenge.roadef.org/2012/en/ (accessed 29 May 2013).

Hansen, P., Mladenović, N., 2001a. Variable neighborhood search. In Pardalos, P.M., Resende, M.G.C. (eds) *Handbook of Applied Optimization*, Oxford University Press, New York.

Hansen, P., Mladenović, N., 2001b. Variable neighbourhood search: principles and applications. *European Journal of Operational Research*, 130, 3, 449–467.

Hussin, M., Stützle, T., 2009. Hierarchical iterated local search for the quadratic assignment problem, In Blesa, M.J., Blum, C., Gaspero, L., Roli, A., Sampels, M., Schaerf, A. (eds) *Hybrid Metaheuristics*, Springer-Verlag, Berlin, Heidelberg, pp. 115–129.

Jansen, K., Öhring, S., 1997. Approximation algorithms for time constrained scheduling. *Information and Computation* 132, 2, 85–108.

Lourenço, H., Martin, O., Stützle, T., 2003. Iterated local search. In Glover, F., Kochenberger, G., Hillier, F. (eds) *Handbook of Metaheuristics, Vol. 57. International Series in Operations Research and Management Science*, Springer, New York, pp. 320–353.

Masson, R., Vidal, T., Michallet, J., Penna, P.H.V., Petrucci, V., Subramanian, A., Dubedout, H. 2013. An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. *Expert Systems with Applications* 40, 5266–5275.

Mehta, D., O'Sullivan, B., Simonis, H., 2012. Comparing solution methods for the machine reassignment problem. In Milano, M. (ed.) *Principles and Practice of Constraint Programming. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 782–797.

Ribeiro, C., Aloise, D., Noronha, T., Rocha, C., Urrutia, S., 2008. An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem. *European Journal of Operational Research* 191, 3, 596–611.

Stützle, T., 2006. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research* 174, 1519–1539.