Appendix A

As mentioned in Section 3.1, the formulation in (P1) is non-linear, due to the last term in the objective function. Nevertheless, it can be linearized by introducing a set of additional variables ξ_n, representing the minimum deployment cost on node $n \in \mathcal{N}$, and γ_{jn}, corresponding to the deployment cost to run job j on node n until its execution is completed. The linearized formulation reads:

$$\min \sum_{j \in \mathcal{J}} (\omega_j \tau_j + \rho \omega_j \hat{\tau}_j) + \mu \sum_{\substack{n \in \mathcal{N} \\ v \in \mathcal{V}}} \left(G_v y_{nv} - \sum_{\substack{j \in \mathcal{J} \\ g \in \mathcal{G}_v}} g x_{jnvg} \right) + \sum_{n \in \mathcal{N}} \xi_n \tag{P1a}$$

subject to:

(P1b) - (P1s) and:

$$\xi_n \geq \sum_{j \in \mathcal{J}} \gamma_{jn} \qquad \forall n \in \mathcal{N} \tag{P1b}$$

$$\gamma_{jn} \leq \pi_{jn} \qquad \forall j \in \mathcal{J},\ \forall n \in \mathcal{N} \tag{P1c}$$

$$\gamma_{jn} \leq \widehat{M}_j^c \alpha_{jn} \qquad \forall j \in \mathcal{J},\ \forall n \in \mathcal{N} \tag{P1d}$$

$$\pi_{jn} - \widehat{M}_j^c (1 - \alpha_{jn}) \leq \gamma_{jn} \qquad \forall j \in \mathcal{J},\ \forall n \in \mathcal{N} \tag{P1e}$$

$$\xi_n \geq 0 \qquad \forall n \in \mathcal{N} \tag{P1f}$$

$$\gamma_{jn} \geq 0 \qquad \forall j \in \mathcal{J},\ \forall n \in \mathcal{N}. \tag{P1g}$$

Given the Constraints (P1b)-(P1g), it is easy to verify that if $\alpha_{jn} = 1$, then ξ_n equals the deployment cost of the first job $j \in \mathcal{J}$ completing under the selected GPUs assignment. Indeed:

- If $\alpha_{jn} = 1$, then the corresponding Constraint (P1c) is more restrictive than the Constraints (P1d) $\gamma_{jn} \leq \widehat{M}_j^c$ since $\pi_{jn} \leq \widehat{M}_j^c$. Moreover, from the corresponding Constraint (P1e) $\gamma_{jn} \geq \pi_{jn}$, we get $\gamma_{jn} = \pi_{jn}$. Finally, since we have a minimization problem, also Constraints (P1b) hold as equalities and γ_{jn} is equal to the cost to execute job $j \in \mathcal{J}$ completely under the selected GPUs assignment.
- If $\alpha_{jn} = 0$, then Constraint (P1d) entails $\gamma_{jn} \leq 0$. Due to Constraint (P1g) we get $\gamma_{jn} = 0$. Moreover, the corresponding Constraint (P1e) becomes $\gamma_{jn} \geq \pi_{jn} - \widehat{M}_j^c$, which is always satisfied since the right-hand side is negative.

Since we have a minimization problem, also Constraints (P1b) hold as equalities and ξ_n is equal to the only γ_{jn} corresponding to $\alpha_{jn} = 1$, i.e., it amounts to the cost to execute entirely the first job $j \in \mathcal{J}$ that will end under the selected GPUs assignment on node $n \in \mathcal{N}$.

Appendix B

In the following, we detail how the complexity of our proposed method (see Section 3.3) can be computed.

Let J be the cardinality of the list of jobs $\mathcal{J}$, N the cardinality of the set of nodes $\mathcal{N}$, and $C = \sum_{v \in \mathcal{V}} G_v$ the cardinality of the set $\mathcal{V} \times \mathcal{G}_v$, i.e., the total number of available configurations. Finally, let σ denote the number of candidate good-quality solutions saved in the set $\mathcal{S}^*$. The overall complexity of our method can be derived as described in the following.

First of all, the complexity of the preprocessing stage is given by $\mathcal{O}(JC)$, due to the update of all processing times. The postprocessing phase has instead a complexity of $\mathcal{O}(N + J)$. Finally, the complexity of the scheduling process is given by the sum of different terms.

First of all, the randomized construction process determines a term $\mathcal{O}(J \log J)$, due to the sorting operation required to build the list $\mathcal{J}_s$. To support the best-fit approach implemented in the selection of the best configuration, the set D_j^* is built as an ordered multimap. This introduces a complexity of $\mathcal{O}(C \log C)$, but reduces the selection to an $\mathcal{O}(1)$ operation. The assignment process has, for all jobs in $\mathcal{J}$, a complexity of $\mathcal{O}(N \log N)$ in the case of the assignment to an existing node, of $\mathcal{O}(G_{v^*})$ (that can be approximated as $\mathcal{O}(1)$, since G_v is considerably smaller than all the other dimension for all $v \in \mathcal{V}$) in the case of the assignment to a new node, and of $\mathcal{O}(CN)$ for the assignment to a suboptimal configuration. Having determined the configuration for all jobs, a final term $\mathcal{O}(\log \sigma)$ is due to the insertion of the new candidate solution in the set $\mathcal{S}^*$. Similarly to D_j^*, this is defined as an ordered map, so that the candidate solutions are automatically sorted with respect to the relative cost.

Since the randomized construction procedure is repeated MaxIt_{RG} times, its overall complexity is given by:

$$\mathcal{O}\left(\text{MaxIt}_{\text{RG}} \left(J \log J + J \left(C \log C + N \log N + CN\right) + \log \sigma\right)\right).$$

We can assume that C and σ are significantly lower than J and N, therefore this can be reduced to:

$$\mathcal{O}\left(\text{MaxIt}_{\text{RG}} \left(J \log J + JN \log N\right)\right).$$

The path relinking procedure is characterized by a complexity of $\mathcal{O}(JN)$ for what concerns the GET_MOVES function. If M is the number of feasible moves in $\mathcal{M}$, given that the exploration step has a worst-case complexity of $\mathcal{O}(J + JN)$, we have an additional term $\mathcal{O}(MJ + MJN) = \mathcal{O}(MJN)$. It is worth noting that the complexity of the exploration step is reduced, in the average case, to $\mathcal{O}(JN)$, which is due to the recursion. Indeed, the choice of an unordered associative container to represent the schedule assigned to each job entails that the average access cost is $\mathcal{O}(1)$. Since the path relinking procedure is repeated MaxIt_{PR} times, its complexity is:

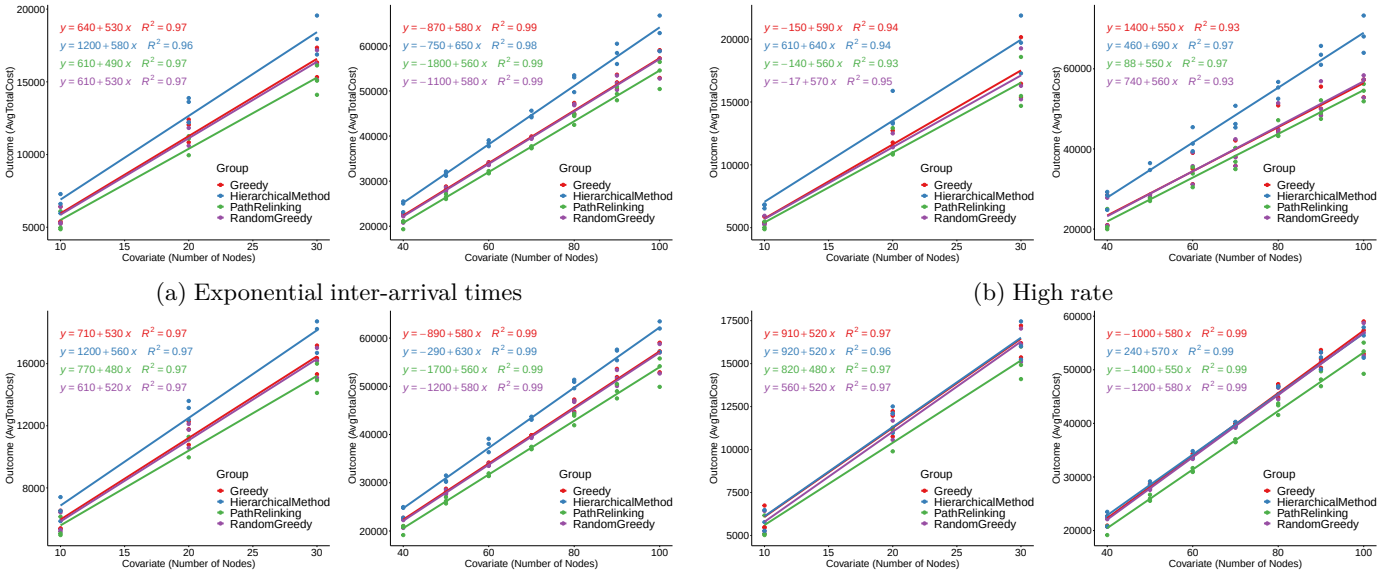
$$\mathcal{O}\left(\sigma\ \text{MaxIt}_{\text{PR}}\ M\ J\ N\right).$$

Thus, neglecting the terms due to preprocessing and postprocessing, that are significantly lower than the others, the overall complexity of our method can be written as:

$$\mathcal{O}\left(\text{MaxIt}_{\text{RG}} \left(J \log J + JN \log N\right) + \sigma\ \text{MaxIt}_{\text{PR}}\ M\ J\ N\right).$$

Appendix C

The results presented in Section 4.3 have been validated statistically, proving the significance of the observed differences among the proposed methods. In particular, we have considered the Analysis of Covariance (ANCOVA) method: it can be used to test the null hypothesis that the means of two or more populations are equal in the presence of a covariate variable, likely to correlate with the dependent variable [31]. In our context, we test the difference in the mean among the total cost obtained with all heuristic methods variants, and the difference in the outcome produced by the Hierarchical Method. Our covariate is the number of nodes in the system, which directly impacts the total cost of an experiment. In particular, we have considered, for each inter-arrival distribution described in Section 4.1, two different scenarios: a small-scale system scenario, characterized by a number of nodes $N \leq 30$, and a large-scale system scenario, identified by $N \geq 40$.



(a) Exponential inter-arrival times

(b) High rate

(c) Mixed rate

(d) Low rate

Figure 1: Hypothesis: linearity between the covariate and the outcome variable in each group, for small and large systems.

The ANCOVA method can be applied under the following hypothesis:

- Linearity between the covariate and the outcome variable at each level of the grouping variable, which has been tested as shown in Figure 1 obtaining a value of R^2 ranging between 0.93 and 0.99 in all considered scenarios.
- Homogeneity of regression slopes, which evaluates that there is no interaction between the outcome and the covariate. This has been checked by performing an ANOVA test [21] on the interaction between outcome and covariate, obtaining, in all considered scenarios, p-values between 0.01 and 0.8.
- Normality of the outcome variable in each group. This has been checked through the Shapiro-Wilk test of normality [32], obtaining p-values between 0.26 and 0.71 in all considered scenarios.
- Homogeneity of variance for all groups. This has been checked through the Bartlett test [33], obtaining, in all considered scenarios, p-values between 0.74 and 0.99.
- No significant outliers in all groups.

The results of the ANCOVA test are reported in Figure 2. We observe statistical significance in all the considered scenarios, with the only exception of the small system with *low* rate. As previously mentioned, a *low* rate in small systems drastically decreases the load, reducing the difference between the various algorithms in terms of solution quality. In general, it is worth observing that the difference between the costs obtained with our heuristic algorithm and with the Hierarchical Method is always more significant in large systems, where the higher load makes it harder to find solutions that minimize due date violations. In these situations, we can observe that the Path Relinking algorithm usually guarantees a solution cost significantly lower than Randomized Greedy and pure Greedy methods. Even if this cannot be observed in smaller systems, we can notice in Figure 2a, Figure 2b and Figure 2c that Path Relinking, Randomized Greedy and pure Greedy obtain significantly lower costs than the Hierarchical Method.

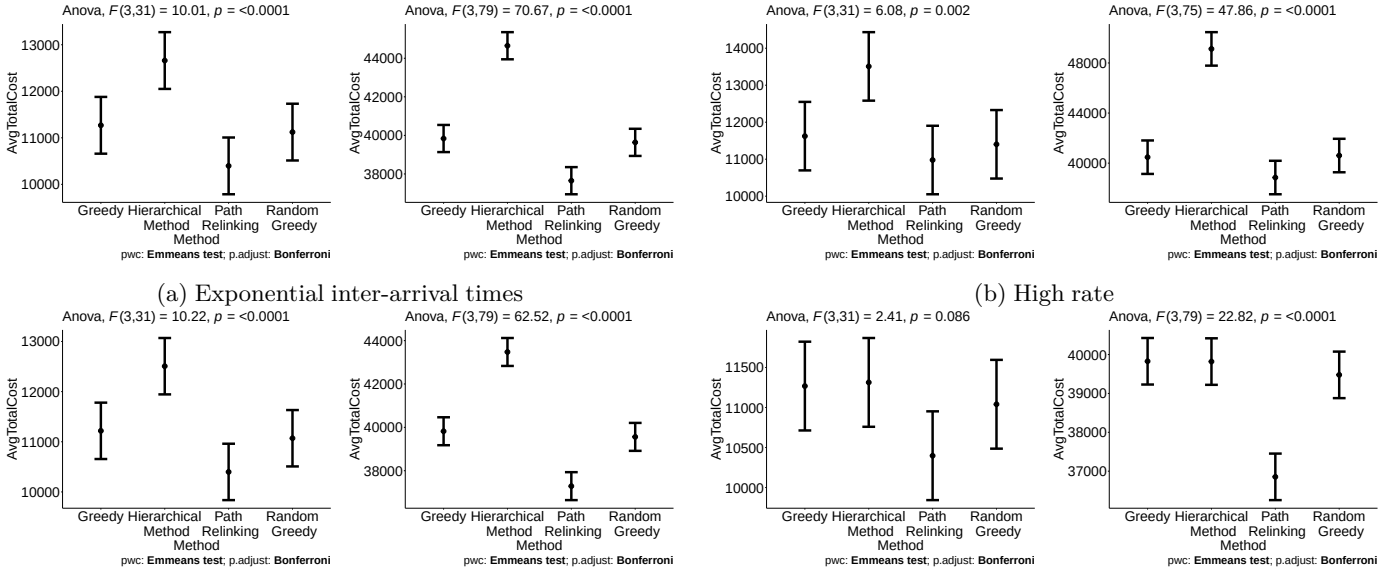
As a final consideration, we did not perform statistical tests to evaluate the significance of the cost reduction obtained by our algorithms with respect to the dynamic programming-based methods presented in Section 4.2.1. Indeed, such methods always yield to higher costs than the Hierarchical Method, which has already been proved to obtain significantly worse performance than our Path Relinking algorithm.

Appendix D

The plots in Figure 3 report the average total costs obtained with the different methods in the setting discussed in Section 4.5. In particular, these scenarios were generated considering problem instances with N available nodes for our methods, while considering instances with the same job traces, but $2N$, $4N$ or $8N$ available nodes for Earliest Deadline First (EDF) and the Dynamic Programming (DP)-based methods presented in Section 4.2.1.

We can observe that the results in the $4N$ and $8N$ scenarios are almost identical, because the number of available nodes in the $4N$ setting is large enough to execute all the concurrent jobs, and no advantages are introduced by considering a larger amount of resources. The DP(AdjWCT) method always yields to lower costs with respect to the other DP-based methods in the $4N$ and $8N$ scenarios. Indeed, the others tend to always select the fastest configuration to execute all jobs, which is not necessary and possibly more expensive when there are enough available resources to run all jobs. In the $2N$ case, a similar pattern can be obseved in the *mixed* and *low* rate settings, where the system load is lower.

Despite the larger amount of resources exploited by EDF and the DP-based methods, Path Relinking yields better results in all the considered scenarios, even if the percentage cost reduction is reduced, in the worst case, from 96 to 10% with respect to EDF and from 39 to 8% against DP(AdjWCT) (see Figure 4).



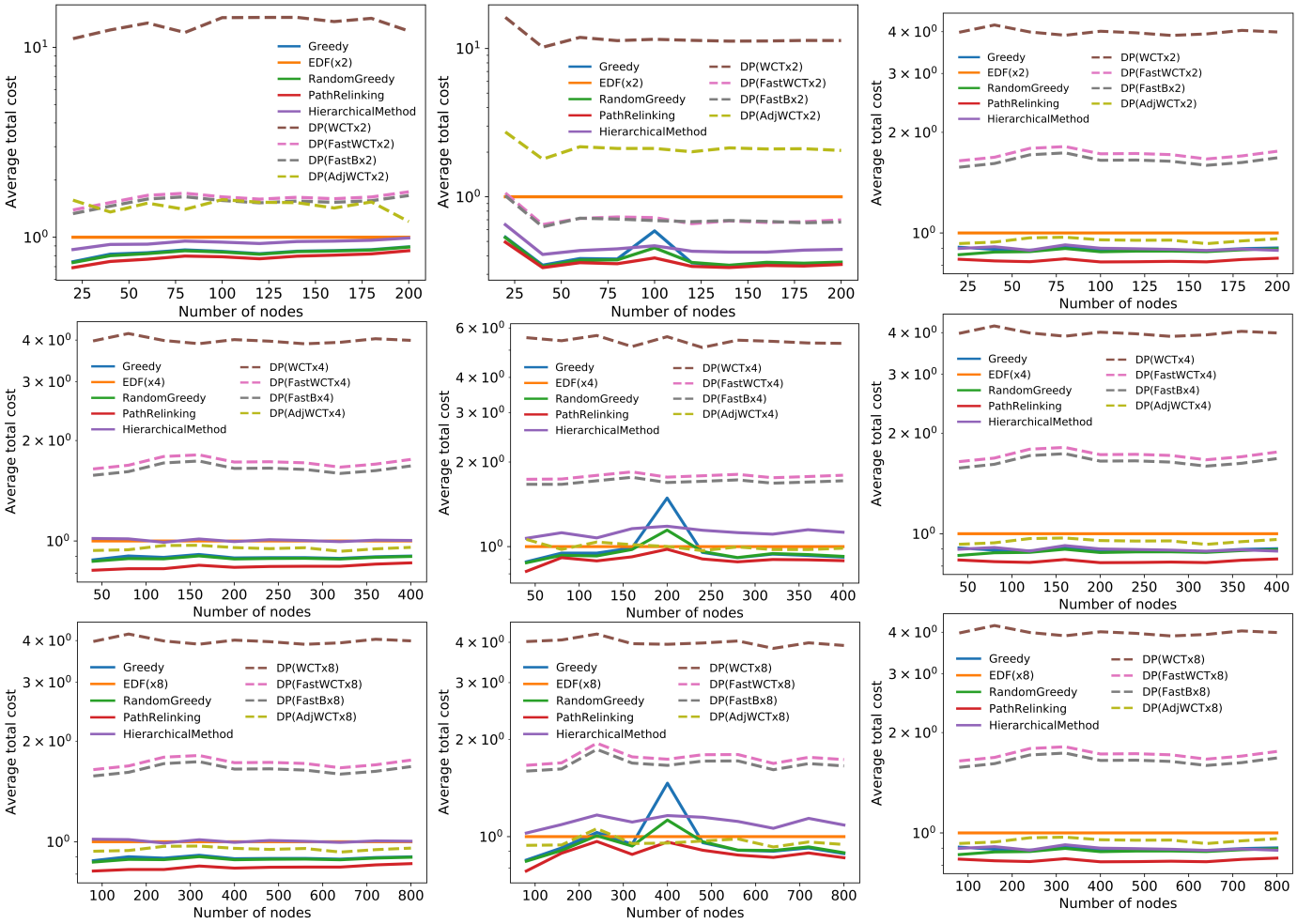
(a) Exponential inter-arrival times

(b) High rate

(c) Mixed rate

(d) Low rate

Figure 2: Results of ANCOVA model for small and large systems

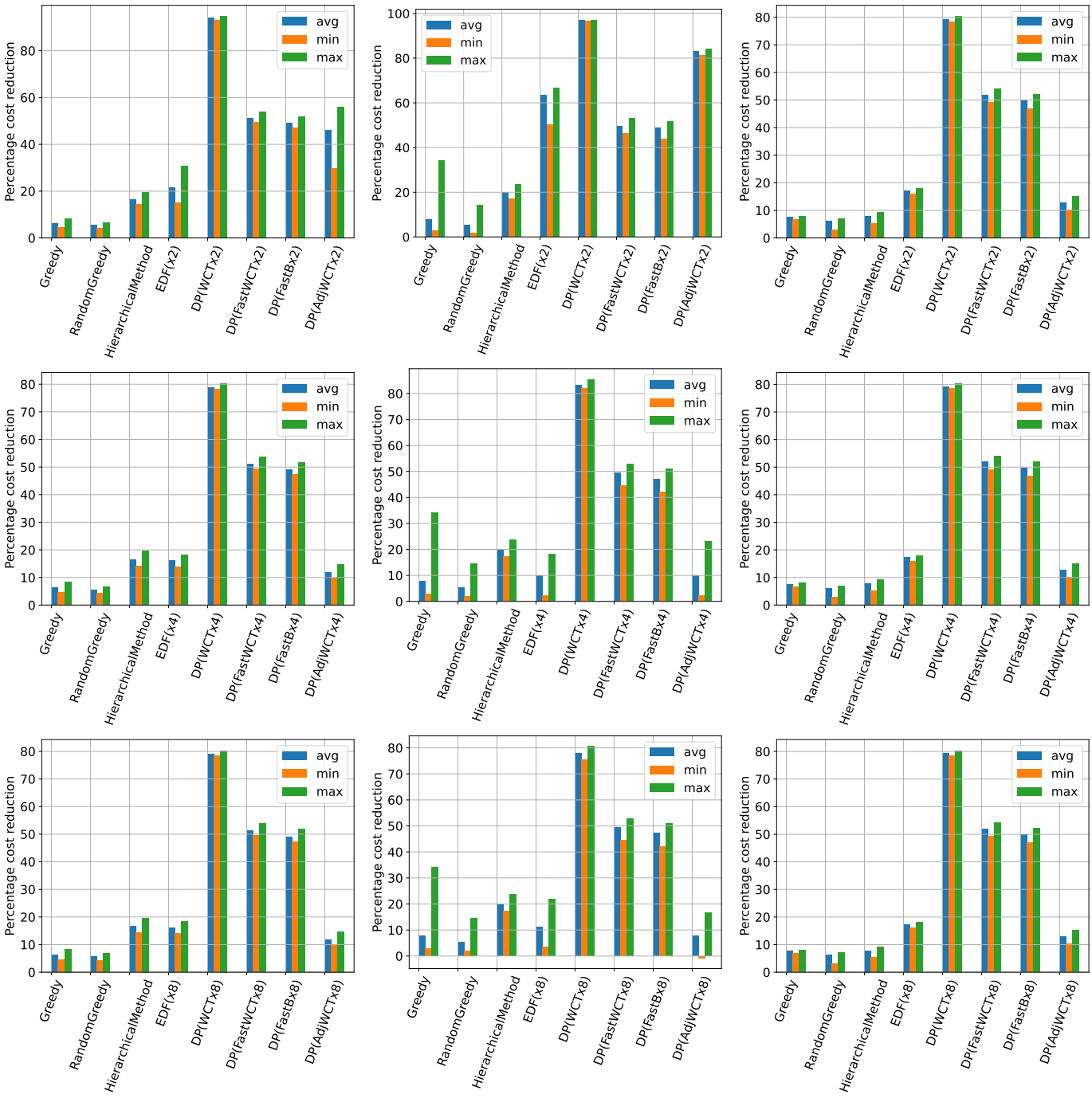


(a) Exponential inter-arrival times

(b) High rate

(c) Low rate

Figure 3: Average total costs comparison in the $2N$, $4N$ and $8N$ scenarios - Note: the number of nodes reported on the x-axis is the one used by EDF and the DP-based methods



(a) Exponential inter-arrival times (b) High rate (c) Low rate

Figure 4: Percentage cost reduction obtained by Path Relinking with respect to the other methods in the $2N$, $4N$ and $8N$ scenarios