Supplementary Materials: Some Multi-Objective Local Search Algorithms Are Better than Others

Anonymous submission

Introduction

This supplementary document provides additional material for the main submission titled *"Some Multi-Objective Local Search Algorithms Are Better than Others"* submitted to the 40th annual AAAI conference on artificial intelligence.

The rest of the document is structured as follows. The first section introduces the multi-objective combinatorial optimisation problems considered in this paper. The second and third sections give the experimental settings and additional experimental results, respectively. The fourth section explains the procedure of investigating the distribution of good neighbours among solutions in the archive. Lastly, the fifth section gives the additional proofs regarding the two toy examples in the main paper.

Multi-Objective Combinatorial Problems

We consider four MOCOPs, the multi-objective 0/1 knapsack (Teghem 1994), travelling salesman problem (TSP) (Ribeiro et al. 2002), quadratic assignment problem (QAP) (Knowles and Corne 2003) and NK-landscapes (Aguirre and Tanaka 2004). Each problem was instantiated in three sizes (100, 200 and 500 decision variables).

Multi-Objective 0-1 Knapsack Problem (Knapsack). The multi-objective 0-1 knapsack problem (Teghem 1994) is a widely studied MOCOP. Given a set of D items $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \{0,1\}^D$, the m-objective problem is defined as the following.

$$\begin{aligned} \max\ f_j(x) = \sum_{i=1}^{D} v_{ji} x_i, \quad j = 1, \ldots, m \\ \text{s.t.} \sum_{i=1}^{D} w_i x_i \leq c \end{aligned} \tag{1}$$

Here, $v_{ji} \geq 0$ is the value of the item i in objective j, w_i is the item's weight, and $c = \frac{1}{2}\sum w_i$ is the capacity. Following (Li et al. 2024), both v_{ji} and w_i are sampled uniformly from $\{10, 11, \ldots, 100\}$.

Multi-Objective Travelling Salesman Problem (TSP). The multi-objective TSP extends the classical TSP, with multiple costs between each pair of cities (Ribeiro et al. 2002), and aims to find the route minimising multiple travelling costs for visiting all the cities exactly once, returning to the start. Formally, given a network $L = (V, C)$, where $V = \{v_1, \ldots, v_D\}$ is a set of D nodes and $C = \{C_j : j \in \{1, \ldots, m\}\}$ is a set of m cost matrices between nodes $(C_j : V \times V)$, the problem is to find the Pareto optimal set of Hamiltonian cycles that minimise each of the m cost objectives.

Multi-Objective Quadratic Assignment Problem (QAP). The multi-objective QAP (Knowles and Corne 2003) models facility-location assignments with multiple flow types. Given m cost matrices $[C_{1,i,j}], \ldots, [C_{m,i,j}]$ and a distance matrix $[L_{u,v}]$, a solution is a permutation $x = (x_1, ..., x_D)$ where x_i denotes the location of facility i. The problem is defined as the following.

$$\min\ f_k(x) = \sum_{i=1}^{D}\sum_{j=1}^{D} C_{k,i,j} L_{x_i,x_j}, \quad k = 1, \ldots, m \tag{2}$$

Multi-Objective NK-Landscape. The multi-objective NK-landscapes (Aguirre and Tanaka 2004) are widely used due to their tunable ruggedness (Verel et al. 2013). Here, N represents the length of the bit-string and K represents the epistasis degree (i.e., each variable is influenced by K other variables, collectively referred to as its locus). For consistency, we denote the length of the bit-string as D. Then, in a m-objective NK-landscape problem, each objective f_j is defined as:

$$\begin{aligned} \max f_j(x) = \frac{1}{D}\sum_{i=1}^{D} c_{ij}(x_i, x_{k_{ij1}}, ..., x_{k_{ijK}}), \\ j = 1, \ldots, m. \end{aligned} \tag{3}$$

Where c_{ij} represent the fitness contribution of the i-th variable, influenced by K other variables in its locus that collectively decide its contribution to the j-th objective. Each c_{ij} depends on the values of the i-variable and the variables in its locus, resulting in 2^{K+1} possible combinations of input and the corresponding output values. Each output is randomly sampled from $(0, 1]$. Following (Aguirre and Tanaka 2007; Daolio et al. 2015), the K other variables of a variable's locus are drawn independently and uniformly at random for each i (variable) and j (objective), resembling a random epistasis pattern.

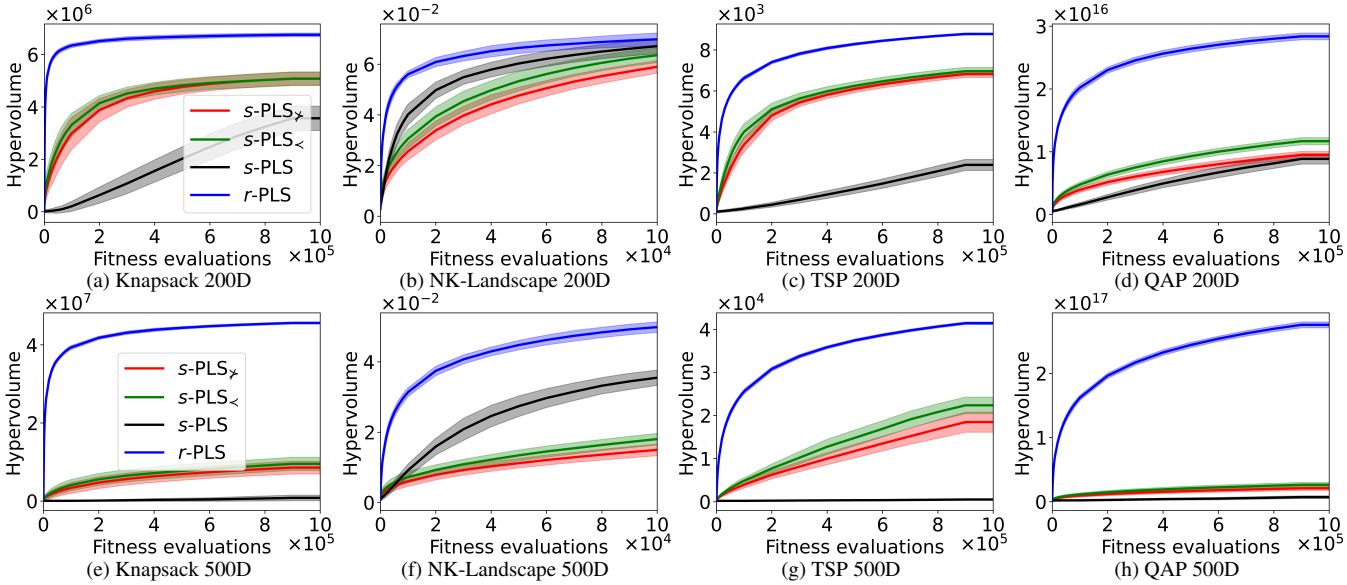


Figure 1: The hypervolume trajectory (higher is better) of the considered s-PLS, s-PLS$_{\not\prec}$, s-PLS$_{\prec}$ and r-PLS across 30 runs on the four MOCOPs with 200 variables (the top panel) and with 500 variables (the bottom panel). The bolded line and shaded area represent the mean and standard deviation of the hypervolume, respectively.

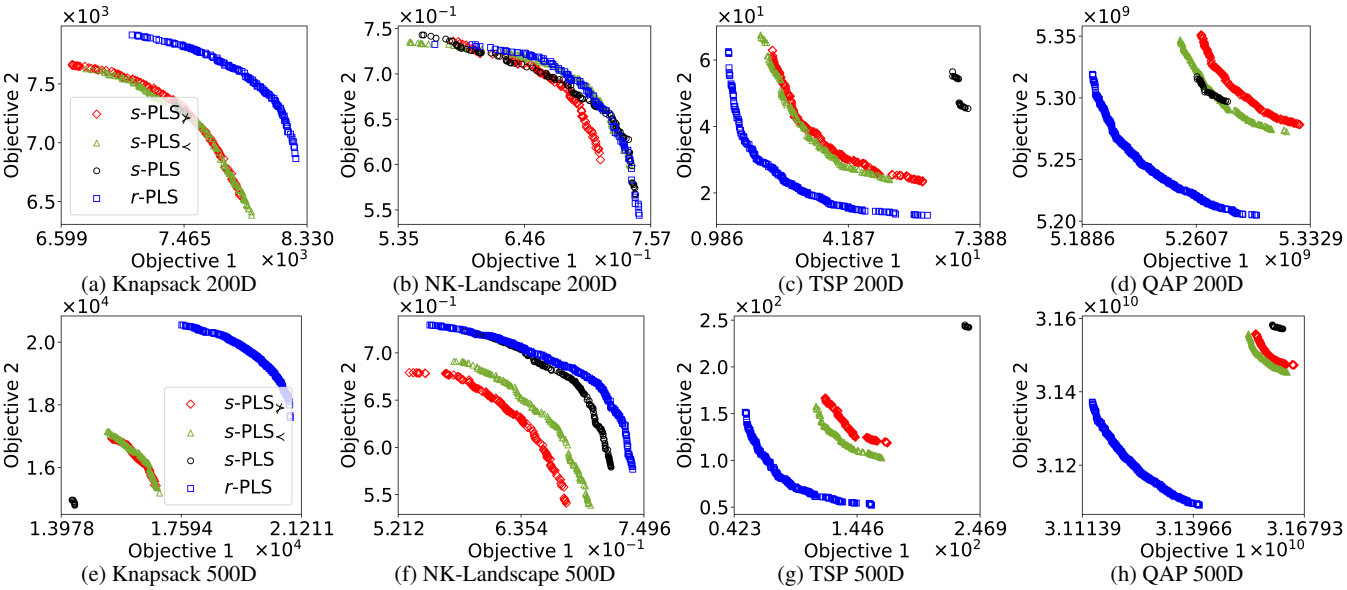


Figure 2: All nondominated solutions (i.e., solutions in the archive) obtained by the considered s-PLS, s-PLS$_{\not\prec}$, s-PLS$_{\prec}$ and r-PLS in a typical run on the four MOCOPs with 200 variables (the top panel) and with 500 variables (the bottom panel), where the Knapsack and NK-Landscape are maximisation problems, and the TSP and QAP are minimisation problems.

Experimental Settings

In our experiments, for each MOCOP we consider three problem sizes (100, 200, and 500). It is important to note that r-PLS and s-PLS typically use different stopping conditions: r-PLS runs until the evaluation budget is exhausted, whereas s-PLS may terminate earlier when there are no more unexplored solutions in the archive. To ensure fairness, all algorithms start from the same initial solution, and r-PLS is forcibly terminated when s-PLS terminates. All algorithms are implemented in Java using the jMetal platform (Durillo and Nebro 2011). For each problem instance, each algorithm is executed over 30 independent runs, with each run performed on a separate core of an AMD Ryzen R9-7945HX CPU, with 16 GB of memory allocated per run.

Neighbourhood operators are matched to the encoding of the variables for each problem: NK-landscape uses 1-bit flip (binary encoding), TSP uses 2-opt (order-based permutation encoding), and QAP uses 2-swap (position-based permutation encoding) (Eiben and Smith 2015). The knapsack problem requires a 2-bit flip neighbourhood, as 1-bit flip often

fail to yield feasible or improved solutions near the constraint boundary, following the practice in this study (Li et al. 2024).

We evaluate performance using the hypervolume (HV) indicator (Zitzler and Thiele 1999), with respect to a reference point estimated via random sampling. Specifically, we generate 100,000 random solutions from the decision space and only retain the non-dominated ones. We then define the reference point by the nadir point of all sampled non-dominated solutions, following the practice in (Li et al. 2024). This is desirable because using the reference point determined by the non-dominated combined set of all generated solutions may easily cause the HV to be zero since in multi-objective combinatorial optimisation different algorithms may have very different "convergence" (i.e., the closeness to the Pareto front) (Li, Chen, and Yao 2020; Li et al. 2024).

Additional Results

In this section, we present the additional experimental results on the four MOCOPs with the problem sizes of 200 and 500.

Figure 1 shows the average hypervolume (bolded line) and standard deviation (shaded area) of the basic *s*-PLS and *r*-PLS, as well as the two *s*-PLS variants across 30 independent runs on the four problems. The top row corresponds to problems with 200 variables, and the bottom row to those with 500 variables. Across all problem types, *r*-PLS consistently achieves higher hypervolume values than all *s*-PLS variants throughout the search. It also attains better final performance. On the NK-landscape problem, the performance advantage of *r*-PLS is relatively smaller, while the other three problems, *r*-PLS demonstrates a clear edge since the very beginning of the search process, particularly as the problem size increases. It is also worth noting that *s*-PLS$\prec$ and *s*-PLS$\not\succ$ (first-improvement variants) are faster than the original *s*-PLS (best-improvement) on most problems (except for the NK-landscape), which is in line with the previous findings (Liefooghe et al. 2012; Dubois-Lacoste, López-Ibáñez, and Stützle 2015).

Figure 2 presents the non-dominated solutions obtained by the four algorithms in a typical run on the four problems when the search ends in the Figure 1. The layout mirrors that of Figure 1, with smaller instances shown in the top row and larger ones in the bottom row. For the Knapsack and NK-Landscape problems (maximisation), better solutions are located toward the top-right corner; for the TSP and QAP (minimisation), the desirable region lies in the bottom-left. As illustrated, *r*-PLS yields solutions with superior convergence and diversity compared to all *s*-PLS variants. The original *s*-PLS, in particular, often exhibits significantly poorer spread across the objective space than the other algorithms.

The above results demonstrate that *r*-PLS achieves better performance in both convergence and diversity, with the advantage becoming more pronounced as the problem size increases.

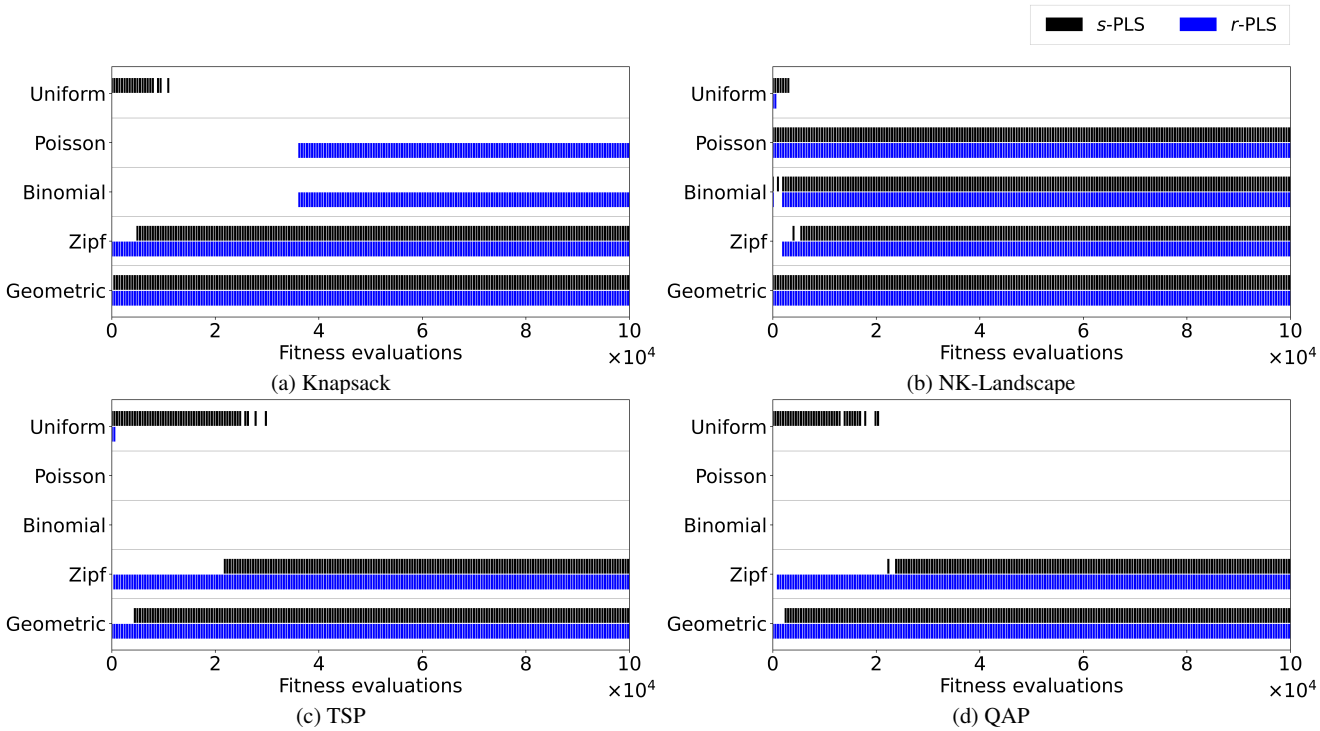
Distribution of Solutions' Neighbours in *s*-PLS and *r*-PLS

This section describes the details on investigating the distribution of the number of good neighbours for *s*-PLS and *r*-PLS. Specifically, we count the number of good neighbours among solutions in the archive during the search process of *s*-PLS and *r*-PLS on the four MOCOPs, and test if this number follows a known discrete probabilistic model. The distribution models (Johnson, Kotz, and Kemp 2005) considered in this paper are the most commonly seen ones, characterised by their different probabilistic tail type: *uniform* (a balance distribution), *Zipf* (heavy-tailed with polynomial decay), *geometric* (light-tailed with exponential decay), *Poisson* (light-tailed with super-exponential decay), and *binomial* (light-tailed with bounded support and a hard cut-off) distributions.

Following the common practice, we evaluate the absolute fit quality using the χ^2 goodness-of-fit test at a significance level of $\alpha = 5\%$. A fit is considered acceptable if the test does not reject the null hypothesis that the observed data come from the tested distribution. Parameters of the distributions were estimated via maximum likelihood. For the Poisson distribution, the rate parameter λ was set to the sample mean. Similarly, for the geometric and binomial distributions, the success probability p was estimated from the sample mean as well. For the Zipf distribution, the exponent parameter s was estimated by numerically minimising the negative log-likelihood over the interval $[1.01, 10]$.

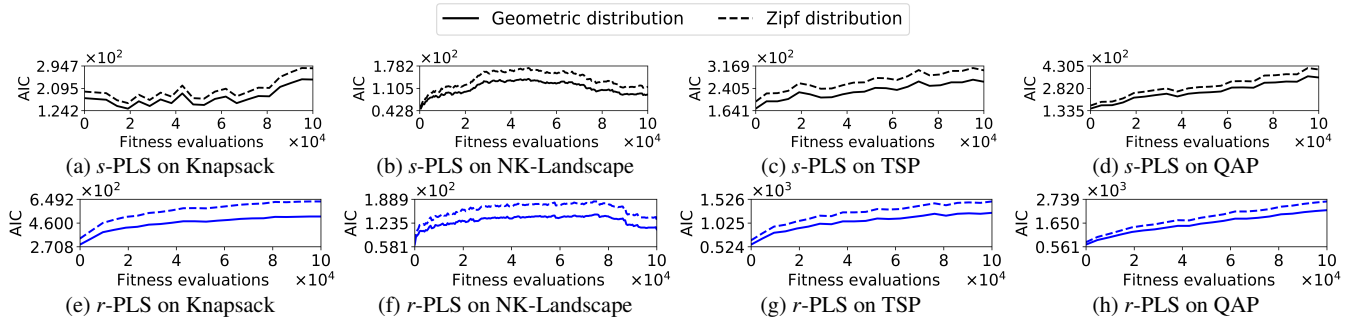
Figure 3 plots the goodness-of-fit of the five discrete distributions with respect to the numbers of good neighbours of solutions in the archive during the search process of *s*-PLS and *r*-PLS on the four MOCOPs. Each horizontal band corresponds to a candidate distribution, and at each sampled evaluation, two coloured ticks, black and blue, respectively indicate a good fit for *s*-PLS and *r*-PLS at $\alpha = 0.05$, namely, the χ^2 test does not reject the distribution. As can be seen in the figure, across nearly all problem instances, the uniform distribution only fits for *s*-PLS at the beginning of the search. In contrast, the geometric distribution passed the χ^2 test almost across all the band of the problems for both *s*-PLS and *r*-PLS, indicating that the number of good neighbours fits well with a geometric distribution.

Note that the Zipf distribution has a comparable number of fits to the geometric distribution across all problems in Figure 3. To compare the relative quality of fit between the two distributions, we use Akaike's Information Criterion (AIC) (Akaike 1974), which balances goodness of fit with model complexity by penalising models with more free parameters. For each distribution, the degrees of freedom for each model are calculated as $n_{\text{bin}} - 1 - k$ where n_{bin} is the number of solutions that have at least one good neighbour and k is the number of free parameters of the distribution. A lower AIC indicates that the model fits better to the data. Figure 4 plots the AIC trajectory of the geometric and Zipf distributions throughout the search processes of *s*-PLS and *r*-PLS on the four problems. The solid line and the dashed line correspond to the geometric and Zipf distributions, respectively. As shown, the solid line lies below the dashed



(a) Knapsack (b) NK-Landscape (c) TSP (d) QAP

Figure 3: Goodness-of-fit of the distributions with respect to the number of good neighbours of solutions in the archive during the search process of *s*-PLS (black) and *r*-PLS (blue) on the (a) Knapsack (100 items), (b) NK-Landscape (N=100, K=10), (c) TSP (100 cities) and (d) QAP (100 factories). A coloured tick in a row indicates that the corresponding algorithm's data at that point was not rejected under the model.



(a) *s*-PLS on Knapsack (b) *s*-PLS on NK-Landscape (c) *s*-PLS on TSP (d) *s*-PLS on QAP

(e) *r*-PLS on Knapsack (f) *r*-PLS on NK-Landscape (g) *r*-PLS on TSP (h) *r*-PLS on QAP

Figure 4: The Akaike's Information Criterion (AIC) (Akaike 1974) trajectories (lower is better) of the geometric distribution (solid line) and the Zipf distribution (dashed line) throughout the search processes of *s*-PLS (top panel) and *r*-PLS (bottom panel) on the four MOCOPs with 100 variables. In all sub-figures, the solid lines lie below the dashed lines, indicating that the geometric distribution consistently provides a better fit than the Zipf distribution.

line in all cases, indicating that the geometric distribution consistently provides a better fit than the Zipf distribution throughout the search process in both *s*-PLS and *r*-PLS.

Proofs of the Two Examples

In the main paper, we present two toy examples of solution's distributions in the archive to help illustrate why *r*-PLS can be faster than *s*-PLS. In Example 1, the archive contains two promising solutions (each having only good neighbours) and two unpromising solutions (each having no good neighbours). In Example 2, all the four solutions in the archive are half-promising (each having an equal mix of good and poor neighbours). Here, we are interested in how quickly (in terms of the number of evaluations) an algorithm can find a good neighbour. In this section, we provide the proofs showing that *r*-PLS is faster in Example 1, whereas *s*-PLS is faster in Example 2.

Proposition 1 (*r*-PLS is faster than *s*-PLS in Example 1)**.** *Let an archive of n solutions contain exactly $n/2$ solutions with no good neighbours ($G = 0$), and the other $n/2$ solutions with all neighbours being good ($G = |\mathcal{N}|$). Then, for any neighbourhood size $|\mathcal{N}| \geq 2$,* r*-PLS requires less time (fewer number of evaluations) than* s*-PLS in finding the next new good solution.*

Proof. For *r*-PLS, the algorithm selects a solution uniformly at random from the archive. With probability $\frac{1}{2}$, it selects a solution whose entire neighbourhood consists of good solutions. Since neighbours are also sampled uniformly, any selected neighbour will be good. Therefore, the success probability is:

$$p_{\text{succ}} = \tfrac{1}{2}, \quad \text{and} \quad \mathbb{E}[T_{r\text{-PLS}}] = \frac{1}{p_{\text{succ}}} = \frac{1}{1/2} = 2. \tag{4}$$

In contrast, *s*-PLS scans the archive sequentially as it marks each visited solution as "explored". For each unpromising solution (one without good neighbours), it evaluates all $|\mathcal{N}|$ neighbours before moving on. This continues until a promising solution is selected from the archive. This process is equivalent to the one described in Lemma 1 (i.e., finding the first good neighbour in the neighbourhood in a sequential scan) of the main paper. Accordingly, the expected number of evaluations required to find first promising solution is:

$$\mathbb{E}[J] = \frac{n+1}{(n/2)+1} = \frac{2(n+1)}{n+2}.$$

The total number of evaluations includes $|\mathcal{N}|$ evaluations for each of the $\mathbb{E}[J]-1$ preceding unpromising solutions, and just 1 evaluation for the first promising one (since all of its neighbours are good). Therefore:

$$\begin{aligned}\mathbb{E}[T_{s\text{-PLS}}] &= \big(\mathbb{E}[J]-1\big)|\mathcal{N}|+1 \\ &= \left(\frac{2(n+1)}{n+2}-1\right)|\mathcal{N}|+1 = \frac{n|\mathcal{N}|}{n+2}+1.\end{aligned} \tag{5}$$

Since $|\mathcal{N}| \geq 2$, it follows that $\mathbb{E}[T_{s\text{-PLS}}] > \mathbb{E}[T_{r\text{-PLS}}]$. □

The above proposition shows that the *r*-PLS is faster than *s*-PLS in finding the first good solution in Example 1. The key reason is that *s*-PLS may waste a substantial number of evaluations when it selects an unpromising solution, as it evaluates all of its neighbours. In contrast, *r*-PLS avoids this by sampling only one neighbour at a time. In Example 2, however, the archive contains no unpromising solutions; instead, all solutions are half-promising – each has an equal mix of good and poor neighbours. In what follows, we show that in Example 2, *s*-PLS is faster than *r*-PLS.

Proposition 2 (*s*-PLS is faster than *r*-PLS in Example 2)**.** *Let the archive consist of n solutions, and suppose that for every solution, exactly half of its neighbours are good; that is, $G = |\mathcal{N}|/2$. Then, for any neighbourhood size $|\mathcal{N}| \geq 2$,* s*-PLS requires less time (fewer number of evaluations) than* r*-PLS in finding the next new good solution.*

Proof. In *r*-PLS, a solution is selected uniformly at random from the archive, and a neighbour is drawn uniformly from its neighbourhood. Since each solution has exactly half good neighbours, the probability of success (i.e., selecting a good neighbour) is:

$$p_{\text{succ}} = \tfrac{1}{2}, \quad \text{and} \quad \mathbb{E}[T_{r\text{-PLS}}] = \frac{1}{p_{\text{succ}}} = \frac{1}{1/2} = 2. \tag{6}$$

As for *s*-PLS, the algorithm explores the neighbourhood of one solution at a time, scanning its neighbours sequentially until a good one is found. Since all solutions in the archive have identical neighbourhood structure, it suffices to consider the expected number of evaluations spent within a single neighbourhood. In this case, *s*-PLS only needs to scan one neighbourhood as all neighbourhoods have $|\mathcal{N}|/2$ good neighbours. By Lemma 1 in the main paper, when half of the neighbours are good, the expected number of evaluations required to find a good neighbour within one neighbourhood is:

$$\mathbb{E}[T_{s\text{-PLS}}] = \mathbb{E}[J] = \frac{|\mathcal{N}|+1}{(|\mathcal{N}|/2)+1} < 2 = \mathbb{E}[T_{r\text{-PLS}}].$$

□

References

Aguirre, H.; and Tanaka, K. 2007. Working principles, behavior, and performance of MOEAs on MNK-landscapes. *European Journal of Operational Research*, 181(3): 1670–1690.

Aguirre, H. E.; and Tanaka, K. 2004. Effects of elitism and population climbing on multiobjective MNK-landscapes. In *Congress on Evolutionary Computation*, volume 1, 449–456.

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6): 716–723.

Daolio, F.; Liefooghe, A.; Verel, S.; Aguirre, H.; and Tanaka, K. 2015. Global vs local search on multi-objective NK-landscapes: contrasting the impact of problem features. In *Genetic and Evolutionary Computation Conference*, 369–376.

Dubois-Lacoste, J.; López-Ibáñez, M.; and Stützle, T. 2015. Anytime Pareto local search. *European Journal of Operational Research*, 243(2): 369–385.

Durillo, J. J.; and Nebro, A. J. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10): 760–771.

Eiben, A. E.; and Smith, J. E. 2015. *Introduction to Evolutionary Computing*. Springer.

Johnson, N. L.; Kotz, S.; and Kemp, A. W. 2005. *Univariate Discrete Distributions*. John Wiley & Sons, 3rd edition.

Knowles, J.; and Corne, D. 2003. Instance generators and test suites for the multiobjective quadratic assignment problem. In *Evolutionary Multi-Criterion Optimization*, 295–310.

Li, M.; Chen, T.; and Yao, X. 2020. How to evaluate solutions in Pareto-based search-based software engineering? A critical review and methodological guidance. *IEEE Transactions on Software Engineering*, 48(5): 1771–1799.

Li, M.; Han, X.; Chu, X.; and Liang, Z. 2024. Empirical Comparison between MOEAs and Local Search on Multi-Objective Combinatorial Optimisation Problems. In *Genetic and Evolutionary Computation Conference*, 547–556. ACM.

Liefooghe, A.; Humeau, J.; Mesmoudi, S.; Jourdan, L.; and Talbi, E.-G. 2012. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18: 317–352.

Ribeiro, C. C.; Hansen, P.; Borges, P. C.; and Hansen, M. P. 2002. A study of global convexity for a multiple objective travelling salesman problem. *Essays and Surveys in Metaheuristics*, 129–150.

Teghem, J. 1994. Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3: 83–104.

Verel, S.; Liefooghe, A.; Jourdan, L.; and Dhaenens, C. 2013. On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives. *European Journal of Operational Research*, 227(2): 331–342.

Zitzler, E.; and Thiele, L. 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4): 257–271.