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"* 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 introduces the procedure of investigating the distribution of the number of good neighbours. Finally, the fifth section gives the additional proofs regarding the two toy examples in the main paper.

Multi-Objective Combinatorial Problems

We consider four commonly used 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. 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.

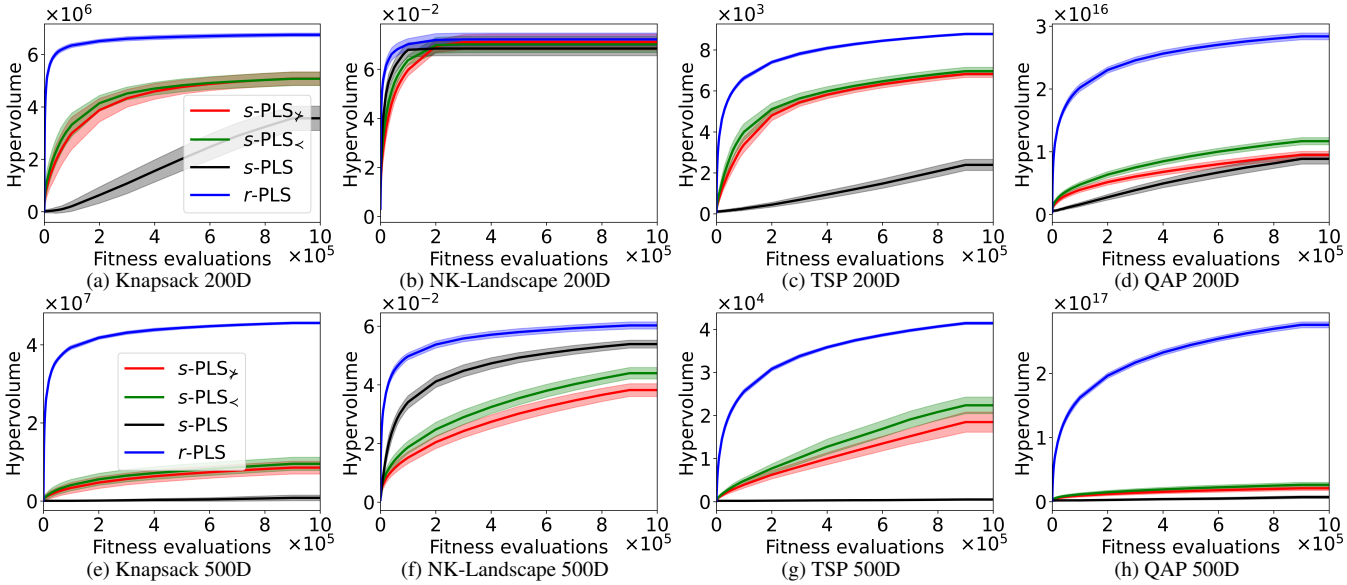


Figure 1: The hypervolume trajectory (the higher the better) of the considered s-PLS, s-PLS$_{\not\prec}$, s-PLS$_{\prec}$ and r-PLS across 30 runs on the four MOCOPs with 200 variables (the top panel) and with 500 variables (the bottom panel). The bolded line and shaded area represent the mean and standard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

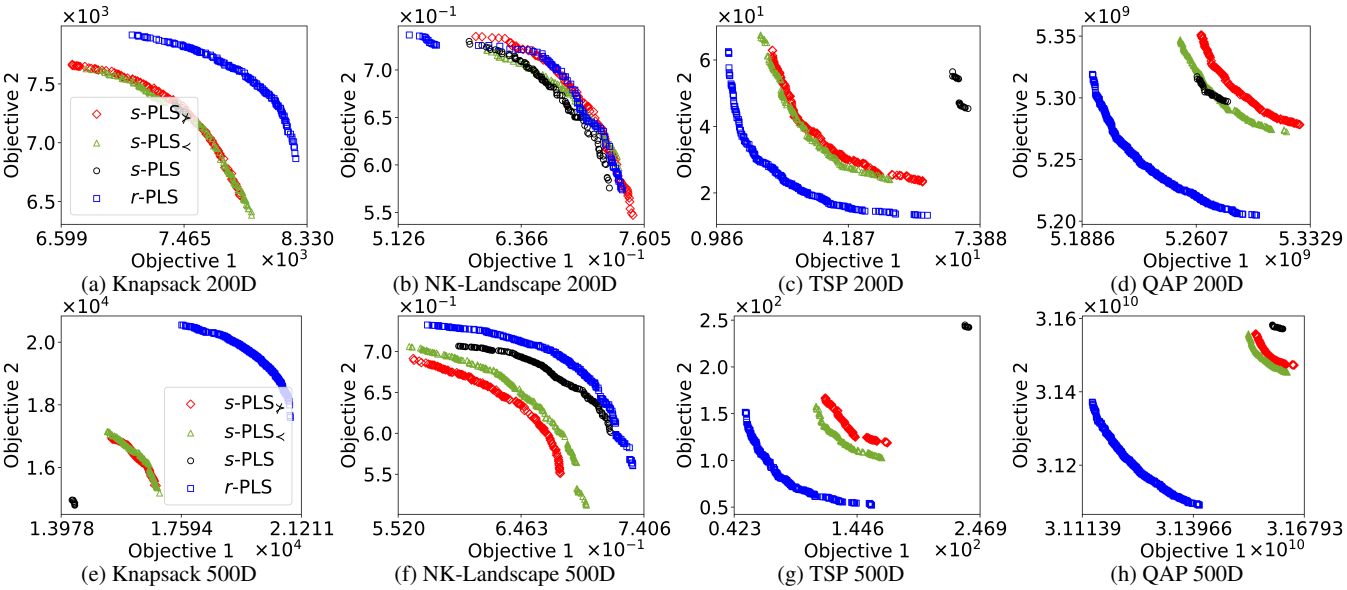


Figure 2: Non-dominated solutions obtained by the considered s-PLS, s-PLS$_{\not\prec}$, s-PLS$_{\prec}$ and r-PLS in a typical run on the four MOCOPs with 200 variables (the top panel) and with 500 variables (the bottom panel), where the Knapsack and NK-Landscape are maximisation problems, and the TSP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

Experimental Settings

In our experiments on the four MOCOPs, we consider four problem sizes (100, 200, and 500), each with a budget of 1,000,000 fitness evaluations. For illustration purposes, the results for the problem size of 100 presented in the main paper are based on only 100,000 fitness evaluations, as s-PLS terminates early once all solutions in the archive are marked as explored.

It is important to note that r-PLS and s-PLS 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). 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.

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 as the point consisting of the worst objective values across all non-dominated sampled solutions, as in the empirical methodology of (Li et al. 2024). This is necessary because using the reference point determined by the non-dominated combined set of all generated solutions may easily cause the HV to be zero (Li, Chen, and Yao 2022; 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 in most cases – except on the NK-Landscape with 200 variables, where all algorithms perform similarly. 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), which aligns with 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 representative run on the four benchmark problems at the end of the search. 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. In contrast, the *s*-PLS algorithms are frequently dominated by *r*-PLS. The original *s*-PLS, in particular, often exhibits significantly poorer spread across the objective space than the other algorithms.

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

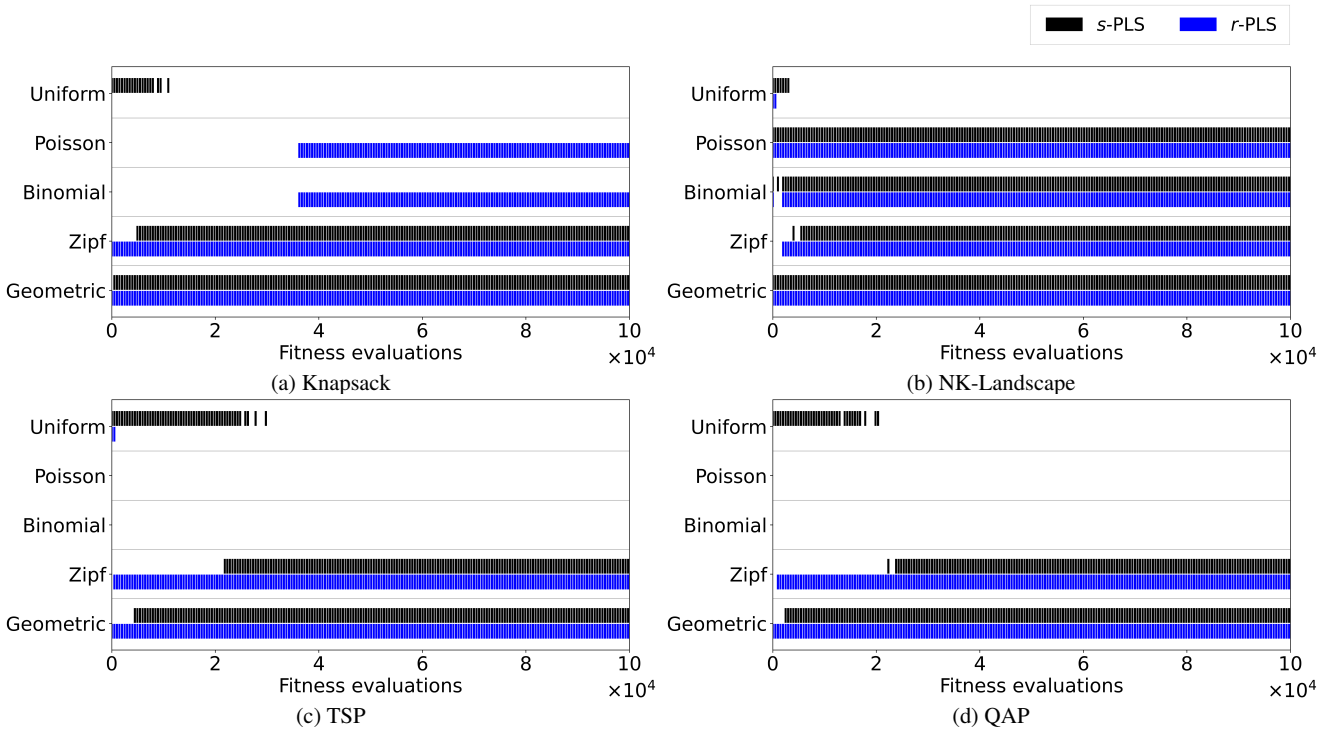
Distribution of Solutions' Neighbours in *s*-PLS and *r*-PLS

This section describes the details in investigating the distribution of the number of good neighbours of the solutions in the archive of *s*-PLS and *r*-PLS during the search. 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. Additionally, we include the categorical distribution as a fallback model, since it can fit any empirical data regardless of underlying structure.

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 $[1.01, 10]$. For categorical distributions, frequencies were derived directly from empirical counts.

To compare the relative quality of fit between distributions, we rank all fits using the Akaike's Information Criterion (AIC) (Akaike 1974). AIC balances goodness of fit with model complexity by penalising distributions 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. Lower AIC indicates a better model that are not overly complex (i.e., less free parameters). If the categorical distribution yields the best AIC, this suggests the absence of a simple generative pattern in the data.

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 and time-scenarios, the geometric distribution passed the χ^2 test more frequently than others, indicating that the number of good neighbours is more consistent with a geometric distribution. In contrast, the uniform distribution is only fitted for *s*-PLS at the beginning of the search. The Poisson distribu-



(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.

tion only fits in the later stages of the search, often alongside the geometric distribution. The binomial distribution only fits on the pseudo-Boolean problems (knapsack and NK-landscape), but it does not fit well on the permutation problems (TSP and QAP). The Zipf distribution achieves a comparable number of acceptable fits to the geometric distribution but is rarely a good fit in the early stages of the search.

As for the relative fit quality, if multiple distributions pass the χ^2 test simultaneously, the geometric distribution also achieved the lowest or the second-lowest AIC by margins often exceeding 10 to the next (i.e. indicate a strong quality difference from the next distribution model). For example, whenever both the geometric and Zipf distributions fit the data, the geometric distribution always yields a lower (better) AIC. Similarly, the binomial distribution typically has the worst AIC on permutation problems (e.g., TSP) and the second-worst on binary problems (e.g., Knapsack). It is worth noting that, although the Poisson distribution passes the goodness-of-fit test in relatively few problems and search stages, they often achieve one of the best AIC scores – often comparable to the geometric distribution – when it does fit.

Proofs of The Two Examples

Proposition 1 (Half good and half bad solutions)**.** *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, assuming $|\mathcal{N}| \geq 2$,* r*-PLS has a smaller expected runtime than* s*-PLS for 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:

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

In contrast, *s*-PLS scans the archive sequentially as it marks each visited solution as "explored". For each unpromising solution (with no good neighbours), it evaluates all $|\mathcal{N}|$ neighbours before moving on. Hence, the expected runtime depends on the position of the first promising solution in the archive. This process is equivalent to the one described in Lemma 1 (i.e., finding the first good neighbour in the neighbourhood) of the main paper, which gives the expected index of the first promising solution as:

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

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 its

neighbours are good). Therefore:

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

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

Proposition 2 (Half neighbours are good for all solutions)**.** *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, assuming $|\mathcal{N}| \geq 2$,* s*-PLS has a smaller expected runtime 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 of its neighbours marked as good, the probability of success (i.e., selecting a good neighbour) is:

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

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 have the same structure, we can analyse the expected runtime by considering just one neighbourhood. Here, the expected runtime equals to the expected index of the first good neighbour when scanning a neighbourhood with $|\mathcal{N}|/2$ good elements. By Lemma 1, this expected runtime 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.

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. 2022. 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*.

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.