DOES FAIRNESS HELP IN MULTI-OBJECTIVE LOCAL SEARCH?

Anonymous authors
Paper under double-blind review

ABSTRACT

Local search, a long-established optimisation technique, has recently attracted renewed attention in multi-objective optimisation. It has been successfully applied to a wide range of practical problems and has recently been shown to outperform well-established evolutionary algorithms. Despite these promising results, the search behaviour of different local search methods, particularly when and why one method is better than another, remains poorly understood. In this paper, we provide both empirical and theoretical insights into this question. We show that local search methods that select solutions more fairly (i.e., according to the number of times they have been visited) generally achieve faster search progress, particularly during the early stages of optimisation. We validate these findings using three local search methods with different levels of fairness on a diverse suite of well-studied problems, including the multi-objective knapsack, travelling salesman, resource allocation, NK-landscape, and quadratic assignment problems.

1 INTRODUCTION

Local search (LS) iteratively modifies a candidate solution within a problem-specific neighbourhood. It is used for SAT (Liang et al., 2025), MaxSAT (Zheng et al., 2025), routing (travelling salesperson and vehicle routing) (Hudson et al., 2022; Ma et al., 2023; Romero Silva et al., 2025), job-shop scheduling (Zhang et al., 2024; Wang et al., 2025a), clustering (Huang et al., 2025), and community detection in signed networks (Aronsson & Haghir Chehreghani, 2025).

Local search has been frequently used in multi-objective optimisation, in which there are more than one objective to optimise at the same time. A key feature in multi-objective optimisation is that there is usually no single optimal solution, but rather a set of Pareto non-dominated solutions, representing different trade-offs between the objectives. In multi-objective local search, a common strategy is to consider the Pareto dominance relation to compare solutions, often referred to as Pareto-based local search (PLS), and use an archive to store non-dominated solutions found during the search. Recently, PLS methods have been shown to outperform well-established multi-objective evolutionary algorithms (MOEAs) on a variety of problems (Phan & Luong, 2023; Li et al., 2024a;b; Li, 2026). The PLS optimisation framework has inspired various variants (the systematic, randomised, decomposition-based, parallel, and adaptive) (Jaszkiewicz, 2018; Shi et al., 2020; 2024). They empirically studied PLS by changing which variation operators are tried, which archive solution is selected, or when a neighbourhood scan stops (Dubois-Lacoste et al., 2015; Antipov et al., 2024; Kang et al., 2024; Wang et al., 2025b).

Yet, the theoretical understanding of search behaviour of PLS methods is rare. In particular, it remains unclear when and why one PLS method is better than another.

It is worth mentioning that in the field of MOEAs, there are plenty of theoretical studies which compute the expected runtime (i.e., evaluations) required of an MOEA to identify the entire Pareto front (Giel, 2003; Laumanns et al., 2004; Doerr & Zheng, 2021; Dinot et al., 2023; Wietheger & Doerr, 2024; Zheng et al., 2024; Bian et al., 2023; Dang et al., 2024; Opris, 2025). Such studies have two notable features:

- They usually analyse the runtime required to identify the entire Pareto front of a multi-objective problem. This does not capture the anytime performance, such as how quickly it discovers high-quality solutions during the search. For example, two algorithms may have

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the same asymptotic runtime for identifying the entire Pareto front, while one consistently finds better solutions at early stages, thereby being preferred given a limited budget.

- The benchmark problems considered are typically simple and synthetic, as their Pareto front must be amenable to theoretical analysis. They often exhibit artificial characteristics, such as linear Pareto fronts, few or no local optima, highly regular fitness landscapes (Liang & Li, 2025), which may not be representative of real-world problems.

In practice, the Pareto front of a multi-objective problem is often computationally intractable, as many multi-objective combinatorial problems, such as Knapsack (Teghem, 1994), TSP (Paquete et al., 2004) and resource allocation problems (Lin & Gen, 2007), are inherently NP-hard (Figueira et al., 2017). As such, practitioners typically care more about how fast an algorithm finds high-quality solutions. In other words, given two methods, which one is better under a given evaluation budget.

Very recently, Liang & Li (2026) made an attempt to analytically investigate two local search methods. They concluded that for PLS randomly choosing one neighbour of an archive solution (called randomised PLS) is faster than systematically scanning the neighbourhood of an archive solution (called systematic PLS), under the assumption that the number of high-quality neighbours of archive solutions follows a geometric distribution.

In this paper, we show this conclusion does not always hold. On some problems, randomised PLS can be slower than systematic PLS during the later stages of the search. Another issue of the work Liang & Li (2026) is the geometric assumption. This specific distribution only covers a limited case – it does not hold at the early stages of the search and different distributions can provide a fit at different stages. In contrast, our work provides general conditions under which one method is faster than others.

More specifically, we aim to investigate when and why one PLS method is faster than another through one concept – *fairness*. We show that choosing archive solutions fairly (i.e., based on the number of times they have been visited) can generally achieve faster progress. However, this advantage is not unconditional. We analyse the conditions under which a fairer PLS method is faster and those under which it is not. We consider three local search methods with different levels of fairness on a diverse suite of widely-studied problems, including the multi-objective knapsack, travelling salesman, resource allocation, NK-landscape, and quadratic assignment problems.

The main contributions of this work are:

- We explain the search behaviours of three commonly used PLS methods from the perspective of fairness, and empirically show that fairer local search is generally faster on a range of multi-objective combinatorial optimisation problems (MOCOPs).
- We derive general conditions under which PLS methods are faster, showing that fairness helps when the chance for finding a non-dominated neighbour is uneven across the archive, but is not always beneficial.
- We prove that, from the same state in the archive, a fairer PLS is no slower in expectation than randomised PLS in finding a non-dominated neighbour.

2 PRELIMINARIES

We consider an m-objective combinatorial optimisation problem over a finite decision space $\mathcal{X}$. For simplicity, we use minimisation notation: $\boldsymbol{f}(x) = (f_1(x), \ldots, f_m(x))$. Given two solutions $x, y \in \mathcal{X}$, x weakly Pareto-dominates y, denoted by $x \preceq y$, if $f_i(x) \leq f_i(y)$, $\forall i \in \{1, \ldots, m\}$. We say x Pareto-dominates y, denoted by $x \prec y$, if $x \preceq y$ and at least one inequality is strict. The Pareto set is the set of solutions that are not dominated by any other solution, and the Pareto front is its image in objective space.

The variation operator in a local search algorithm typically needs a neighbourhood structure. We denote the neighbourhood of a solution x by $\mathcal{N}(x) \subseteq \mathcal{X} \setminus \{x\}$. For example, in a binary problem, $\mathcal{N}(x)$ can be the set of solutions reachable by one bit flip. During the search, a Pareto-based local search maintains an archive A of mutually non-dominated solutions found so far. Throughout the paper, we use the term *good neighbour* to refer to a neighbour of a solution in the archive that is

not dominated by any solution in the archive (i.e., not dominated by any solution discovered by the search so far).

Definition 1 (Good neighbour)**.** Given an archive A, a neighbour $x' \in \mathcal{N}(x)$ is called a *good neighbour* if $\nexists a \in A$ such that $a \preceq x'$.

An archive solution is called *promising* if it has at least one good neighbour, and non-promising otherwise.

Background. Early multi-objective local search often transformed the problem into one or several scalar optimisation problems (Ishibuchi & Murata, 1998; Jaszkiewicz, 2002; Paquete & Stützle, 2003). Pareto-based local search (PLS) instead uses Pareto dominance directly and has become a standard framework for multi-objective local search (Paquete et al., 2004; Blot et al., 2018). It repeatedly selects one solution from the archive, explores its neighbourhood, and updates the archive using Pareto dominance. Many PLS variants differ mainly in two choices: which archive solution is selected, and how the selected solution's neighbourhood is explored. Based on the second choice criterion, there are two basic classes: systematic PLS and randomised PLS.

Systematic PLS (*s*-PLS) scans neighbours of the selected solution one by one. The scan may stop after the whole neighbourhood is exhausted, or may stop earlier when a first-improvement condition is met, such as finding a neighbour that is not dominated with respect to the archive, or one that dominates an archive solution (Liefooghe et al., 2012; Dubois-Lacoste et al., 2015). This class includes the original PLS and later PLS heuristics (Paquete et al., 2004; Dubois-Lacoste et al., 2015; Shi et al., 2024; Kang et al., 2024). Algorithm A.1 in Appendix A.2 gives the general framework of *s*-PLS.

Randomised PLS (*r*-PLS) instead selects an archive solution and samples one of its neighbours uniformly at random. It does not remember which neighbours or archive solutions have already been considered, so it may repeat evaluating the same solutions. This simple randomised form includes SEMO and related variants (Giel, 2003; Laumanns et al., 2004; Drugan & Thierens, 2012; Jaszkiewicz, 2018), and it is also widely used in runtime analysis of evolutionary multi-objective optimisation (Doerr & Zheng, 2021; Dinot et al., 2023; Wietheger & Doerr, 2024). Algorithm A.2 in Appendix A.2 gives the corresponding framework of *r*-PLS.

3 Fairness in Pareto-based Local Search

In this section, we first define what we mean by fairness in PLS and then study how choosing solutions fairly can affect the search behaviour of PLS methods. In this paper, choosing solutions fairly means that evaluations are allocated evenly across solutions in the archive.

Formally, let $C_t(x)$ be the number of evaluations allocated to an archive solution x up to time t. For a fixed archive $\mathcal{A} = \{x_1, \ldots, x_n\}$, we can define the evaluation budget allocation vector after t evaluations as

$$C(t) = (C_t(x_1), C_t(x_2), \ldots, C_t(x_n)). \tag{1}$$

Then, a perfectly fair allocation over a fixed archive satisfies

$$\max_i C_t(x_i) - \min_i C_t(x_i) \leq 1. \tag{2}$$

This leads to fair Pareto-based local search (f-PLS), which visits the least visited archive solution, effectively in a round-robin manner. Algorithm A.3 (in Appendix A.2) gives the framework of f-PLS (e.g., FEMO (Laumanns et al., 2004)). Note that fairly choosing solutions to explore is not a new concept in the area. They have been frequently studied in theoretical works in the MOEA field (Laumanns et al., 2004; Friedrich et al., 2011; Osuna et al., 2020; Antipov et al., 2024).

Systematic PLS, randomised PLS, and fair PLS together form three levels of fairness. Suppose an archive has n solutions, and each neighbourhood $\mathcal{N}(s)$ has N neighbours; here $N = |\mathcal{N}(s)|$ denotes the neighbourhood size.

- *s*-PLS is the least fair of the three. It selects an archive solution and scans its neighbourhood systematically. When *s*-PLS eventually finds a good neighbour of an archive solution (this solution costs $0 < J \leq N$ evaluations to explore its neighbourhood), every archive solution



(a) Example 1 (b) Example 2

Figure 1: Two toy examples illustrating the distribution of good neighbours among solutions in the archive. A filled circle represents a solution for which all neighbours are good. An empty circle represents a solution with no good neighbours. A quarter-filled circle represents a solution for which one quarter of its neighbours are good. In Example 1, only one solution has good neighbours. In Example 2, all four solutions have the same number of good neighbours.

considered before costs N evaluations, and all the remaining archive solutions receive no evaluation.

- *r*-PLS is fairer than *s*-PLS. It selects an archive solution uniformly at random and then samples one neighbour of that solution. Thus, all archive solutions receive the same number of evaluations in expectation. However, in a real run, some archive solutions may be selected many times while others are not selected at all. This is effectively a coupon-collector scenario (Grimmett & Stirzaker, 2001), in which touching all archive solutions requires $\Theta(n \log n)$ evaluations in expectation. Note that the mechanism of Pareto-based local search may cause *r*-PLS not very fair, as archive solutions found earlier naturally receive more evaluations.
- f-PLS, on the other hand, by definition, touches all n solutions in the archive after n evaluations unless a non-dominated neighbour is found. As a result, f-PLS tends to prioritise newly added archive solutions, as they have received fewer evaluations.

We use two toy examples in Figure 1 to illustrate how different levels of fairness affect the cost (the number of evaluations) at which a PLS finds the next good neighbour. In Figure 1(a), there are four solutions in the archive. Three of them have no good neighbours, whereas the remaining one has only good neighbours. In this case, the least fair *s*-PLS has a $3/4$ chance of wasting N evaluations immediately – it expects $1.5N + 1$ evaluations to find the next good neighbour. The less unfair *r*-PLS expects 4 evaluations to select the one promising neighbourhood and find a good neighbour. The explicitly fair f-PLS is even faster – its round-robin style guarantees finding a good neighbour within 4 evaluations, and expects 2.5 evaluations on average.

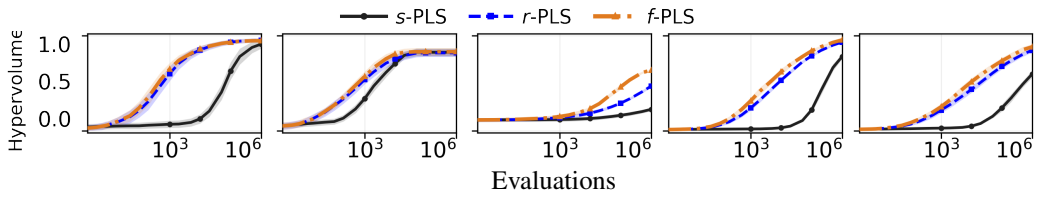
However, fairness does not always help. In Figure 1(b), one quarter of each solution's neighbours are good. When the good neighbours are perfectly evenly distributed across the archive, the order of accessing the solutions in the archive no longer matters. Thus *r*-PLS and f-PLS behave the same – both expect 4 evaluations to find the next good neighbour. On the other hand, regardless of which solution *s*-PLS picks, the expected number of evaluations to find a good neighbour is $\frac{N+1}{N/4+1}$, which is smaller than 4. The proofs of the two examples are in Appendix D.1 and Appendix D.2.

As shown in the two examples, the benefit of fairness depends on how good neighbours are distributed across archive solutions. When good neighbours are sparse and unevenly located, fairness prevents the algorithm from spending too much budget on the unpromising archive solutions. When good neighbours are evenly distributed, fairness gives no advantage, and systematic scanning can be slightly faster in finding the next good neighbour.

4 Empirical Study

We now examine whether the fairness principle translates into faster anytime progress on widely studied multi-objective combinatorial optimisation problems (MOCOPs). We compare the systematic PLS (*s*-PLS), randomised PLS (*r*-PLS), and fair PLS (f-PLS). All algorithms use the same archive update rule and differ only in how fairly they choose the archive solutions.

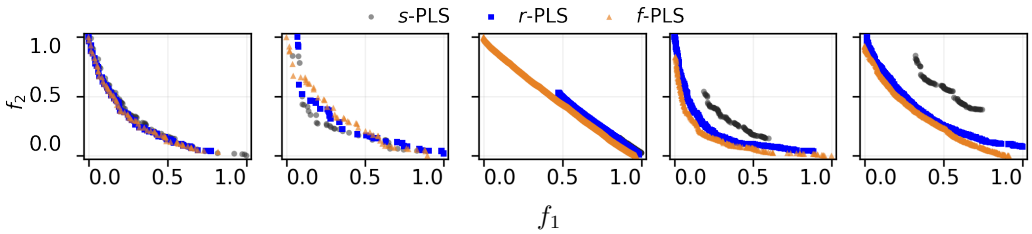
Experimental setup. We consider five MOCOPs: the multi-objective 0/1 knapsack problem (Teghem, 1994), NK-landscape with $K = 10$ (Aguirre & Tanaka, 2004), travelling salesman problem (TSP) (Ribeiro et al., 2002), quadratic assignment problem (QAP) (Knowles & Corne, 2003), and multi-objective resource allocation problem (RAP) (Lin & Gen, 2007). These problems cover constrained and unconstrained binary encodings, integer encodings, and both order-based and



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

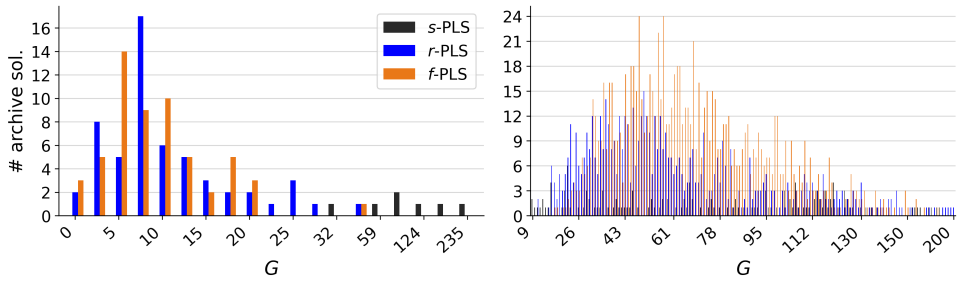
Figure 2: Normalised hypervolume (higher is better) of s-PLS, r-PLS, and f-PLS on the five multi-objective problems with 100 variables. Curves and shaded areas show the mean and standard deviation, respectively, over 30 instances.

position-based permutation encodings. For each setting, we generate 30 instances and run each algorithm once per instance for 10^6 fitness evaluations. We report normalised hypervolume (HV; higher is better) (Zitzler & Thiele, 1999), computed after converting every objective to minimisation. Detailed problem definitions, neighbourhoods, instance-generation procedures, early-termination handling, and HV reference-point settings are provided in Appendix A.1 and Appendix A.3.



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

Figure 3: Representative non-dominated sets obtained at 10^6 evaluations on the five 100-variable problems. Objectives are scaled to $[0, 1]$ minimisation; lower-left is better.



(a) Knapsack (b) RAP

Figure 4: Good-neighbour counts G across archive solutions at 10^5 evaluations on 100-variable knapsack and RAP.

Experimental results. Figure 2 shows the mean normalised HV (curves) and standard deviation (shaded areas) across 30 instances of the five problems with 100 variables. As can be seen, the fairer algorithms (f-PLS and r-PLS) typically progress faster (i.e., having a higher HV curve) than s-PLS in the early and middle stages on all problems. The same pattern can also be observed for larger problem sizes (see Appendix Figure A.1 and Appendix A.4), where s-PLS progresses even slower. This means selecting solutions fairly in a random manner (r-PLS) already helps. Explicit fairness (f-PLS) gives further speed-up: f-PLS (orange) is best at 10^4 evaluations on all five problems, and at 10^6 evaluations it remains the best on RAP, TSP and QAP. The advantage is particularly large on RAP, where at 10^6 evaluations f-PLS reaches a mean normalised HV of 0.638, compared with 0.470

for *r*-PLS and 0.227 for *s*-PLS. On the two permutation problems, f-PLS also keeps an advantage at 10^6 evaluations. Figure 3 shows the solutions obtained by the three PLS methods in a representative run for each of the five problems when the search ends at 10^6 evaluations. As shown, f-PLS finds better solutions (lower-left is better) on RAP, TSP and QAP, while *s*-PLS clearly falls behind on TSP and QAP.

On the other hand, *s*-PLS can progress quickly in some cases. On the knapsack and NK-landscape problems (Figure 2(a) and (b), respectively), *s*-PLS catches up at the later stages of the search. In the lower-right region of Figure 3(a), s-PLS obtains several additional non-dominated solutions. A clearer pattern can be seen with the first-improvement variants of *s*-PLS: they are generally faster and even progress as quickly as the fairer PLS methods on NK-landscape (100D) (see Appendix Figure A.3).

To see why these PLS variants can behave differently across problems, Figure 4 shows the distribution of good neighbours over archive solutions at an intermediate stage of search (after 10^5 evaluations). On RAP, where f-PLS is clearly the fastest at the stage, the good-neighbour counts are spread over a much wider range. This indicates a highly uneven opportunity landscape: some archive solutions have many good neighbours, while many others have far fewer (closer to the first toy example in Figure 1(a)). On the knapsack problem, the archive state of *s*-PLS is small and occupies only a few good-neighbour count categories (closer to the second toy example in Figure 1(b)); this helps explain why its HV can grow rapidly once it reaches promising archive solutions, and why the methods become close later in the run. The next section formalises the comparison between the three PLS methods, by analysing how the distribution of good neighbours affects the expected time to find a new solution that is not dominated by any solution found previously.

5 Theoretical Explanation

The experiments show that fairer PLS methods are often faster, but on some problems (e.g., knapsack, NK-landscape), at later stages of the search, the less fair *s*-PLS can still make quick progress. In this section, we try to further explain this by analysing when and how fairer PLS methods can be faster.

In this analysis, we assume the same fixed archive state for all PLS methods and compare how quickly they find the next good neighbour. Let $G \in \{0, 1, \ldots, N\}$ be the number of good neighbours of a uniformly selected archive solution. We model G as an arbitrary categorical distribution. We write

$$q = \Pr(G = 0), \quad r = 1 - q, \quad \mu = \mathbb{E}[G], \quad \mu^+ = \mathbb{E}[G \mid G > 0]$$

where q is the proportion of non-promising solutions (they have no good neighbour), r is the proportion of promising solutions (they have at least one good neighbour), μ is the average number of good neighbours over the whole archive, and μ^+ is the corresponding average excluding the non-promising solutions. The value of q measures the risk of selecting a non-promising solution (no good neighbours), whereas the conditional distribution of G captures the distribution of good neighbours and is affected by how unevenly the good neighbours are distributed.

We first compare *s*-PLS with *r*-PLS.

Proposition 1 (Expected number of evaluations to find the next good neighbour)**.** *Assume $0 \le q < 1$ and $\mu > 0$. Let $T_{r\text{-}PLS}$ and $T_{s\text{-}PLS}$ be the numbers of evaluations until the next good neighbour is found. Under the same archive,*

$$\mathbb{E}[T_{r\text{-}PLS}] = \frac{N}{\mu}, \tag{3}$$

$$\mathbb{E}[T_{s\text{-}PLS}] = \left(\frac{n+1}{nr+1} - 1\right)N + (N+1)\mathbb{E}\left[\frac{1}{G+1} \mid G > 0\right]. \tag{4}$$

The first expression (*r*-PLS) is straightforward. One evaluation of *r*-PLS succeeds with probability $\mathbb{E}[G/N] = \mu/N$, so the expected number of evaluations needed is N/μ.

The second expression (*s*-PLS) contains two costs. First, the first-position lemma (Appendix C.1) gives $\frac{n+1}{nr+1}$ as the expected position of the first promising solution. Each solution before this position ($\frac{n+1}{nr+1} - 1$) cost a full scan of N neighbours. Second, once it reaches a solution with $G = k > 0$,

Table 1: Comparison of *s*-PLS and *r*-PLS for an arbitrary discrete distribution of good neighbours G. Here n denotes the archive size, N the neighbourhood size, and a promising solution has $G > 0$. The thresholds are defined in Theorems 1 and 2.

Case	Condition	Results
Same number of good neighbours	$G = g,\ g \in \{1, \dots, N\}$	*s*-PLS is no slower, and is faster unless $g = N$, as every selected solution is promising and scanning does not repeat neighbours (Corollary 1).
All promising, small mean	$q = 0,\ \mu \leq 2$	*s*-PLS is no slower for every distribution with this mean (Corollary 2). Random sampling can repeat neighbours when good neighbours are scarce, while systematic scanning cannot.
All promising, large mean	$q = 0,\ 2 < \mu < N$	Across suitable finite archives with the same mean, either method can be faster. Homogeneous good-neighbour counts favour *s*-PLS, whereas heterogeneous counts can favour *r*-PLS (Corollary 3).
Some non-promising, large mean	$0 < q < 1,\ \mu \geq \mu_{\text{high}}(q, n)$	*r*-PLS is guaranteed to be no slower. This is the regime in which committing N evaluations to a non-promising solution is especially costly (Theorem 1).
Some non-promising, very small mean	$N \geq 2,\ 0 < q < 1,\ \mu \leq \mu_{\text{low}}(q, n)$	*s*-PLS is guaranteed to be no slower. The threshold reduces to 2 when $q = 0$ and $N \geq 2$ (Theorem 2).
Stagnation	$q = 1$	All three algorithms are stuck, as there is no good neighbour (Corollary 4).

the expected position of the first good neighbour is $\frac{N+1}{k+1}$.These are respectively the first and second terms of Eq. (4). The complete proof is given in Proposition C.1 in Appendix C.2.

The contrast between the two equations is important. For *r*-PLS, the mean μ contains all the information needed for the expected time to find the next good neighbour. For *s*-PLS, two archives with the same n, q and μ can still give different expected times because the second term in Eq. (4) depends on the conditional distribution of G given $G > 0$. The following bounds show the effect of this distribution.

Lemma 1 (Bounds for systematic PLS)**.** *With Proposition 1,*

$$\frac{nqN}{nr+1} + \frac{N+1}{\mu^+ + 1} \leq \mathbb{E}[T_{\textit{s-PLS}}] \leq \frac{nqN}{nr+1} + 1 + \frac{N-\mu^+}{2}. \tag{5}$$

The lower bound represents the situation favoured by *s*-PLS in which the good-neighbour counts are distributed as evenly as possible among promising solutions. The upper bound represents the opposite: a highly uneven situation, with the counts concentrated towards the extremes. Formally, $1/(G+1)$ is convex, thus, by Jensen's inequality (Boyd & Vandenberghe, 2004), for fixed n, q and μ, a mean-preserving spread of the positive good-neighbour counts cannot decrease the expected time of *s*-PLS. In words, the performance of *s*-PLS is sensitive to both the average number of good neighbours and where they are located. The complete proof is given in Lemma C.1 in Appendix C.3.

5.1 Between Randomised and Systematic PLS: When Is One Faster?

The two bounds in Eq. (5) yield sufficient conditions for comparing *s*-PLS and *r*-PLS. For compactness, write

$$A := \frac{nqN}{nr+1}, \qquad C := r - \frac{Nq}{nr+1}.$$

Here A is the expected evaluation cost of scanning non-promising solutions before the first promising one.

Theorem 1 (When randomised PLS is faster)**.** *Assume $0 < q < 1$ and $\mu > 0$. If*

$$\mu \geq \mu_{\text{high}}(q, n) := \frac{-C + \sqrt{C^2 + 4ANr}}{2A},$$

then $\mathbb{E}[T_{r\text{-}PLS}] \leq \mathbb{E}[T_{s\text{-}PLS}]$.

Theorem 2 (When systematic PLS is faster)**.** *Assume* $N \geq 2$*,* $0 \leq q < 1$ *and* $\mu > 0$*, and let* $B = 2rA + r(N+2)$*. If*

$$\mu \leq \mu_{\text{low}}(q,n) := \frac{B - \sqrt{B^2 - 8Nr}}{2},$$

then $\mathbb{E}[T_{s\text{-}PLS}] \leq \mathbb{E}[T_{r\text{-}PLS}]$.

Theorem 1 describes the setting in which good neighbours are sufficiently abundant on average, but some archive solutions have none. Here the main factor is that *s*-PLS commits an entire scan to a non-promising solution, whereas *r*-PLS is not. The proof compares $\mathbb{E}[T_{r\text{-PLS}}]$ with the lower bound for *s*-PLS in Lemma 1, and then solves the resulting quadratic inequality in μ. The complete proof is given in Theorem C.1 in Appendix C.4.

Theorem 2 describes the opposite setting, in which good neighbours are sufficiently scarce. Here, scanning a neighbourhood (sampling without replacement) is valuable because repeated neighbour evaluations are comparatively expensive. The proof compares the upper bound for *s*-PLS in Lemma 1 with $\mathbb{E}[T_{r\text{-PLS}}]$, and again solves the resulting quadratic inequality in μ. The complete proof is given in Theorem C.2 in Appendix C.5. When $\mu_{\text{low}}(q,n) < \mu < \mu_{\text{high}}(q,n)$, these sufficient conditions do not determine the ordering; the exact comparison uses the conditional distribution of G.

The thresholds depend on the archive size as well as q and N. For example, take $N = 4950$ and an archive of $n = 100$ solutions. If $q = 0.1$, then $\mu_{\text{low}} \approx 1.64$ and $\mu_{\text{high}} \approx 2.87$; if $q = 0.5$, they are approximately 0.68 and 0.72. Thus, even a mean of a few good neighbours can suffice to favour *r*-PLS. Conversely, sufficiently scarce good neighbours favour *s*-PLS under Theorem 2. These conditions suggest possible mechanisms for the empirical behaviour, but do not by themselves establish the ordering of hypervolume trajectories from evolving, algorithm-specific archives. For fixed q, $r > 0$ and N, the failure cost $nqN/(nr+1)$ tends to Nq/r as $n \to \infty$.

The comparison between *s*-PLS and *r*-PLS also includes the following edge cases.

Corollary 1 (All neighbourhoods are identical)**.** *If* $G = g$ *for every archive solution, where* $1 \leq g \leq N$*, then* s*-PLS is faster than* r*-PLS, except for the trivial case* $g = N$.

Substituting $q = 0$ and $\mu = \mu^+ = g$ into Proposition 1 gives $(N+1)/(g+1) \leq N/g$, with equality only when $g = N$. This is the case in the second toy example (Figure 1(b)), where *s*-PLS is guaranteed to find a good neighbour in one neighbourhood. The complete proof is given in Corollary C.1 in Appendix C.6.

Corollary 2 (All promising and small mean)**.** *If* $q = 0$ *and* $\mu \leq 2$*, then* s*-PLS is no slower than* r*-PLS.*

This follows by setting $q = 0$ in the upper bound of Lemma 1 and comparing it with $\mathbb{E}[T_{r\text{-PLS}}] = N/\mu$. The complete proof is given in Corollary C.2 in Appendix C.6.

Corollary 3 (All promising and large mean)**.** *For* $q = 0$ *and* $2 < \mu < N$*, then either algorithm can be faster. If* $\mu = N$*, then the two algorithms are equal.*

Here, we compare N/μ with both the lower and upper bounds for *s*-PLS: constant or adjacent good-neighbour counts favour systematic scanning, whereas a wide distribution favours fairer methods. The complete proof is given in Corollary C.3 in Appendix C.6.

Corollary 4 (Stagnation)**.** *If* $q = 1$*, then* $G = 0$ *always. In this fixed state, all three algorithms need infinite expected time to find the next good neighbour.*

This is the trivial case in which no solution has a good neighbour. All search algorithms are stuck at this stage.

All cases are summarised in Table 1. Intuitively, the decision depends on how many solutions are non-promising and how unevenly the remaining opportunities are distributed. This connects the model to the empirical distributions in Figure 4 of the main paper, and Figure B.1 of the appendix, where the distributions of good neighbours on all considered problems are given.

5.2 From Random Fairness to Explicit Fairness

Although *r*-PLS is fair in expectation, it often results in an unfair allocation of evaluations in practice, as solutions that enter the archive early may be selected many times, whereas newly added solutions may be selected only a few times. We next compare *r*-PLS with the even fairer f-PLS. For a fixed archive of n solutions, let p_i be the probability that one random neighbour of archive solution i is good. Then, $p_i = G_i/N$ and $\bar{p} = n^{-1}\sum_{i=1}^{n} p_i$. *r*-PLS selects archive solutions with replacement and has an expected time of $1/\bar{p}$ when $\bar{p} > 0$; f-PLS instead selects every solution one time before any is selected again.

Proposition 2 (Fair PLS is no slower than randomised PLS)**.** *For any* $t \in \mathbb{Z}_{\geq 0}$, $\Pr(T_{f\text{-}PLS} > t) \leq \Pr(T_{r\text{-}PLS} > t)$, *thus*

$$\mathbb{E}[T_{f\text{-}PLS}] \leq \mathbb{E}[T_{r\text{-}PLS}].$$

In words, at any moment, f-PLS is less likely to be still looking for a good neighbour. The proof compares sampling archive solutions without replacement within a fair round against repeated sampling with replacement, using Maclaurin's inequality (Hardy et al., 1988). The complete proof is given in Proposition C.2 in Appendix C.7. The advantage grows as the good-neighbour counts become more heterogeneous. This aligns with the wide distribution observed for RAP in Figure 4.

Note that the analysis is a *time-to-next* comparison under a fixed search state and therefore does not tell which algorithm obtains better solutions by the end of the search. In practice, the distribution of good neighbours changes continuously, and such neighbours become scarcer as the search progresses. Fairer methods may therefore reach these more difficult search states sooner, as they progress more quickly. However, even though the distribution favours fairer methods only in a period, this period is often long enough for them to find many good solutions. Indeed, on most problems, this advantage persists throughout 10^6 evaluations, particularly for larger problem sizes (see Appendix Figure A.1).

6 Conclusion

This paper shows empirically and analytically that fairness generally improves multi-objective local search when good neighbours are unevenly distributed, whereas systematic search can be faster when they are scarce or evenly distributed. We recommend fair allocation when uneven opportunities are likely. Future work could distinguish good neighbours by their effects rather than treating them as equally useful.

AI use statement

For preparation of paper, generative AI models (ChatGPT) was used to search for related papers, polish the manuscript and to check formatting requirements of ICLR. The authors take responsibility for the final content of the submission, including any AI-assisted formatting.

References

Hernán E Aguirre and Kiyoshi Tanaka. Effects of elitism and population climbing on multiobjective mnk-landscapes. In *CEC*, pp. 449–456, 2004.

Denis Antipov, Timo Kötzing, and Aishwarya Radhakrishnan. Greedy versus curious parent selection for multi-objective evolutionary algorithms. In *PPSN*, pp. 86–101, 2024.

Linus Aronsson and Morteza Haghir Chehreghani. An Efficient Local Search Approach for Polarized Community Discovery in Signed Networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Chao Bian, Yawen Zhou, Miqing Li, and Chao Qian. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *IJCAI*, pp. 5513–5521, 2023.

Aymeric Blot, Marie-Éléonore Kessaci, and Laetitia Jourdan. Survey and unification of local search techniques in metaheuristics for multi-objective combinatorial optimisation. *Journal of Heuristics*, 24(6):853–877, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. ISBN 978-0-521-83378-3.

Duc-Cuong Dang, Andre Opris, and Dirk Sudholt. Level-Based Theorems for Runtime Analysis of Multi-objective Evolutionary Algorithms. In *PPSN*, pp. 246–263, 2024.

Matthieu Dinot, Benjamin Doerr, Ulysse Hennebelle, and Sebastian Will. Runtime analyses of multi-objective evolutionary algorithms in the presence of noise. In *IJCAI*, pp. 5549–5557, 2023.

Benjamin Doerr and Weijie Zheng. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *AAAI*, pp. 12293–12301, 2021.

Mădălina M Drugan and Dirk Thierens. Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics*, 18:727–766, 2012.

Jérémie Dubois-Lacoste, Manuel López-Ibáñez, and Thomas Stützle. Anytime pareto local search. *Eur. J. Oper. Res.*, 243(2):369–385, 2015.

Juan J. Durillo and Antonio J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011. ISSN 0965-9978.

Agoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*. Springer, 2015.

José Rui Figueira, Carlos M Fonseca, Pascal Halffmann, Kathrin Klamroth, Luís Paquete, Stefan Ruzika, Britta Schulze, Michael Stiglmayr, and David Willems. Easy to say they are hard, but hard to see they are easy—towards a categorization of tractable multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):82–98, 2017.

Tobias Friedrich, Christian Horoba, and Frank Neumann. Illustration of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 412(17):1546–1556, 2011.

O. Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *CEC*, pp. 1918–1925, 2003.

Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 3rd edition, 2001.

G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, Cambridge, 2 edition, 1988.

Junyu Huang, Zhen Zhang, Beirong Cui, Jianxin Wang, and Qilong Feng. Fast local search algorithms for clustering with adaptive sampling and bandit strategies. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph Neural Network Guided Local Search for the Traveling Salesperson Problem. In *International Conference on Learning Representations*, 2022.

Hisao Ishibuchi and Tadahiko Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans. Syst., Man, Cybern.*, 28(3):392–403, 1998.

Andrzej Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Trans. Evol. Comp.*, 6(4):402–412, 2002.

Andrzej Jaszkiewicz. Many-objective pareto local search. *Eur. J. Oper. Res.*, 271(3):1001–1013, 2018.

Yuhao Kang, Jialong Shi, Jianyong Sun, Qingfu Zhang, and Ye Fan. New techniques to improve neighborhood exploration in Pareto local search. *Expert Syst. Appl.*, 254:124296, 2024. ISSN 0957-4174.

Joshua Knowles and David Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In *EMO*, pp. 295–310, 2003.

Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, 1997.

M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Trans. Evol. Comput.*, 8(2):170–182, 2004.

Miqing Li. Combinatorial optimisation can be different from continuous optimisation for MOEAs. In *GECCO Tutorial*, pp. 1167–1181, 2026.

Miqing Li, Xiaofeng Han, Xiaochen Chu, and Zimin Liang. Empirical comparison between MOEAs and local search on multi-objective combinatorial optimisation problems. In *GECCO*, pp. 547–556, 2024a.

Yan Li, Yundu Huang, Wuyang Mao, Furong Ye, Xiang He, Zhonglin Zu, and Shaowei Cai. Bi-objective contract allocation for guaranteed delivery advertising. In *KDD*, pp. 1691–1700, 2024b.

Jiaxin Liang, Junping Zhou, and Minghao Yin. Diversat: A novel and effective local search algorithm for diverse SAT problem. In *AAAI*, pp. 11290–11298, 2025.

Zimin Liang and Miqing Li. On the problem characteristics of multi-objective pseudo-Boolean functions in runtime analysis. In *FOGA*, pp. 166–177, 2025.

Zimin Liang and Miqing Li. Random is faster than systematic in multi-objective local search. In *AAAI*, pp. 37072–37080, 2026.

Arnaud Liefooghe, Jérémie Humeau, Salma Mesmoudi, Laetitia Jourdan, and El-Ghazali Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18:317–352, 2012.

Chi-Ming Lin and Mitsuo Gen. Multiobjective resource allocation problem by multistage decision-based hybrid genetic algorithm. *Applied Mathematics and Computation*, 187(2):574–583, 2007. ISSN 0096-3003.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to Search Feasible and Infeasible Regions of Routing Problems with Flexible Neural k-Opt. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Andre Opris. A first runtime analysis of NSGA-III on a many-objective multimodal problem: Provable exponential speedup via stochastic population update. In *IJCAI*, pp. 8903–8911, 2025.

Edgar Covantes Osuna, Wanru Gao, Frank Neumann, and Dirk Sudholt. Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*, 832:123–142, 2020.

Luis Paquete and Thomas Stützle. A two-phase local search for the biobjective traveling salesman problem. In *EMO*, pp. 479–493, 2003.

Luis Paquete, Marco Chiarandini, and Thomas Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In *Metaheuristics for Multiobjective Optimisation*, pp. 177–199. 2004.

Quan Minh Phan and Ngoc Hoang Luong. Pareto local search is competitive with evolutionary algorithms for multi-objective neural architecture search. In *GECCO*, pp. 348–356, 2023.

Celso C Ribeiro, Pierre Hansen, Pedro Castro Borges, and Michael Pilegaard Hansen. A study of global convexity for a multiple objective travelling salesman problem. In *Essays and Surveys in Metaheuristics*, pp. 129–150. Springer, 2002.

Andrés Felipe Romero Silva, Alejandra Tabares Pozos, David Álvarez-Martínez, and John Willmer Escobar. An iterated local search based approach for a real rich vehicle routing problem with time windows. *Computers & Industrial Engineering*, 208:111422, 2025.

Jialong Shi, Qingfu Zhang, and Jianyong Sun. PPLS/D: Parallel Pareto local search based on decomposition. *IEEE Trans. Cybern.*, 50(3):1060–1071, 2020. ISSN 2168-2275.

Jialong Shi, Jianyong Sun, Qingfu Zhang, Haotian Zhang, and Ye Fan. Improving pareto local search using cooperative parallelism strategies for multiobjective combinatorial optimization. *IEEE Trans. Cybern.*, 54(4):2369–2382, 2024.

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

Jiaqi Wang, Zhiguang Cao, Peng Zhao, Rui Cao, YuBin Xiao, YUAN JIANG, and You Zhou. Learning Memory-Enhanced Improvement Heuristics for Flexible Job Shop Scheduling. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Wei Wang, Jialong Shi, Jianyong Sun, Arnaud Liefooghe, and Qingfu Zhang. A new parallel cooperative landscape smoothing algorithm and its applications on TSP and UBQP. *Expert Systems with Applications*, 263:125611, 2025b.

S. Wietheger and B. Doerr. Near-tight runtime guarantees for many-objective evolutionary algorithms. In *PPSN*, pp. 153–168, 2024.

Cong Zhang, Zhiguang Cao, Wen Song, Yaoxin Wu, and Jie Zhang. Deep Reinforcement Learning Guided Improvement Heuristic for Job Shop Scheduling. In *International Conference on Learning Representations*, 2024.

Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, Chu-Min Li, and Felip Manyà. Integrating multi-armed bandit with local search for maxsat. *Artificial Intelligence*, 338:104242, 2025.

W. Zheng, M. Li, R. Deng, and B. Doerr. How to use the metropolis algorithm for multi-objective optimization? In *AAAI*, pp. 20883–20891, 2024.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comp.*, 3(4):257–271, 1999.

APPENDIX

A EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

A.1 MULTI-OBJECTIVE COMBINATORIAL OPTIMISATION PROBLEMS

We consider five multi-objective combinatorial optimisation problems (MOCOPs): the multi-objective 0/1 knapsack problem (Teghem, 1994), NK-landscape with $K = 10$ (Aguirre & Tanaka, 2004), multi-objective resource allocation problem (RAP) (Lin & Gen, 2007), travelling salesman problem (TSP) (Ribeiro et al., 2002) and quadratic assignment problem (QAP) (Knowles & Corne, 2003). We consider three problem sizes for each problem: 100, 200 and 500 decision variables.

Multi-Objective 0/1 Knapsack Problem (Knapsack). The multi-objective 0/1 knapsack problem (Teghem, 1994) is a constrained binary optimisation problem. Given D items and a binary solution $\mathbf{x} = (x_1, \ldots, x_D) \in \{0, 1\}^D$, the bi-objective problem is

$$\max \; f_j(\mathbf{x}) = \sum_{i=1}^{D} v_{ji} x_i, \quad j = 1, 2, \quad \text{s.t.} \sum_{i=1}^{D} w_i x_i \leq c \tag{A.1}$$

where, v_{ji} is the value of item i in objective j, w_i is the item's weight, and $c = \frac{1}{2}\sum_i w_i$ is the capacity. In the generated instances, each value and weight is sampled uniformly from $\{10, 11, \ldots, 100\}$.

Multi-Objective NK-Landscape. The NK-landscape (Aguirre & Tanaka, 2004) is a pseudo-Boolean benchmark with tuneable ruggedness. Here D denotes the bit-string length, and K denotes the number of other bits that interact with each bit. In our experiments, $K = 10$ for all three problem sizes. For each objective j, the objective value is

$$\max \; f_j(\mathbf{x}) = \frac{1}{D} \sum_{i=1}^{D} c_{ij}\left(x_i, x_{k_{ij1}}, \ldots, x_{k_{ijK}}\right), \quad j = 1, 2. \tag{A.2}$$

For every variable and objective, the K interacting variables are sampled independently and uniformly at random, without choosing the variable itself. Each possible combination of c_{ij} is sampled uniformly from $[0, 1)$.

Multi-Objective Resource Allocation Problem (RAP). The RAP is an integer-quantity resource allocation problem (Lin & Gen, 2007). A solution is $\mathbf{x} = (x_1, \ldots, x_D)$, where x_i is the number of resource units assigned to activity i. In our generated instances,

$$x_i \in \{0, 1, \ldots, 20\}, \qquad \sum_{i=1}^{D} x_i \leq B,$$

where $B = 10D$. The first objective minimises the total cost, and the second objective maximises the total benefit:

$$\min \; f_1(\mathbf{x}) = \sum_{i=1}^{D} c_i(x_i), \qquad \max \; f_2(\mathbf{x}) = \sum_{i=1}^{D} b_i(x_i). \tag{A.3}$$

The generated cost function c and benefit function d for an activity i monotonically increase when more resource units are allocated. For each additional resource unit, the cost increment is sampled from $\{3, \ldots, 15\}$, and the benefit increment is sampled from $\{4, \ldots, 20\}$.

Multi-Objective Travelling Salesman Problem (TSP). The multi-objective TSP extends the classical TSP to multiple cost matrices (Ribeiro et al., 2002). Given D cities and two symmetric cost matrices C_1 and C_2, a solution is a Hamiltonian cycle. The objective is to minimise both tour costs:

$$\min \; f_j(\mathbf{x}) = \sum_{i=1}^{D} C_j(x_i, x_{i+1}), \qquad j = 1, 2, \tag{A.4}$$

where $x_{D+1} = x_1$. In the generated instances, each objective has an independently generated symmetric random cost matrix, where each value is sampled from [0,1).

Multi-Objective Quadratic Assignment Problem (QAP). The multi-objective QAP (Knowles & Corne, 2003) models facility-location assignments with multiple flow matrices. Let L be a distance matrix between locations and let C_1, C_2 be two flow matrices between facilities. A solution is a permutation $\mathbf{x} = (x_1, \ldots, x_D)$, where x_i is the location assigned to facility i. The problem is

$$\min \; f_j(\mathbf{x}) = \sum_{i=1}^{D} \sum_{\ell=1}^{D} L_{i\ell} C_{j,x_i,x_\ell}, \qquad j = 1, 2. \tag{A.5}$$

The generated instances use a Euclidean distance matrix (each value is sampled from [0,5000]) and two random flow matrices (each value is sampled from [0,100]), following (Liang & Li, 2026).

A.2 Algorithmic Frameworks

For completeness, Algorithms A.1–A.3 give the detailed pseudocode for the three PLS variants compared in the main paper. The systematic and randomised baselines follow the frameworks studied in Liang & Li (2026).

Algorithm A.1 Systematic Pareto-based Local Search (*s*-PLS)

Require: Evaluation budget T and neighbourhood function $\mathcal{N}$
Sample $x \in \mathcal{X}$ */* $\mathcal{X}$: the decision space */*
$A \leftarrow \{x\}, U \leftarrow \{x\}, t \leftarrow 1$ */* U: unexplored archive solutions */*
while $t < T$ **and** $U \neq \varnothing$ **do**
  $x \sim \mathrm{Unif}(U)$ */* Select an unexplored solution */*
  repeat
    $y \leftarrow next_neighbour(\mathcal{N}(x))$
    if $\nexists a \in A : a \preceq y$ **then**
      $A \leftarrow (A \cup \{y\}) \setminus \{a \in A \mid y \prec a\}$ */* Update the non-dominated archive */*
      $U \leftarrow (U \cap A) \cup \{y\}$
    end if
    $t \leftarrow t + 1$
  until $stop_condition()$ */* e.g., exhaustion or first improvement; or $t = T$ */*
  $U \leftarrow U \setminus \{x\}$
end while
return A

Algorithm A.2 Randomised Pareto-based Local Search (*r*-PLS)

Require: Evaluation budget T and neighbourhood function $\mathcal{N}$
Sample $x \in \mathcal{X}$ */* $\mathcal{X}$: the decision space */*
$A \leftarrow \{x\}, t \leftarrow 1$
while $t < T$ **do**
  $x \sim \mathrm{Unif}(A)$ */* Select a solution from the archive */*
  $y \sim \mathrm{Unif}(\mathcal{N}(x))$
  if $\nexists a \in A : a \preceq y$ **then**
    $A \leftarrow (A \cup \{y\}) \setminus \{a \in A \mid y \prec a\}$ */* Update the non-dominated archive */*
  end if
  $t \leftarrow t + 1$
end while
return A

Algorithm A.3 Fair Pareto-based Local Search (f-PLS)

Require: Evaluation budget T and neighbourhood function $\mathcal{N}$

Sample $x \in \mathcal{X}$ /* $\mathcal{X}$: the decision space */
$A \leftarrow \{x\}, C(x) \leftarrow 0, t \leftarrow 1$ /* $C(x)$: number of neighbour evaluations allocated to x */
while $t < T$ **do**
  $x \sim \mathrm{Unif}(\arg\min_{a \in A} C(a))$ /* Select a least visited solution from the archive */
  $y \sim \mathrm{Unif}(\mathcal{N}(x))$
  $C(x) \leftarrow C(x) + 1$
  if $\nexists a \in A : a \preceq y$ **then**
    $A \leftarrow (A \cup \{y\}) \setminus \{a \in A \mid y \prec a\}$ /* Update the non-dominated archive */
    $C(y) \leftarrow 0$
  end if
  $t \leftarrow t + 1$
end while
return A

A.3 Experimental Settings

In the experiments, we consider systematic PLS (s-PLS, instantiated by the original Pareto local search (Paquete et al., 2004)), randomised PLS (r-PLS, instantiated by SEMO (Laumanns et al., 2004)), and fair PLS (f-PLS, instantiated by FEMO (Laumanns et al., 2004)). For each problem and each problem size, we conduct 30 runs and each run is assigned with a random independently generated instance. For comprehensiveness, we additionally include two first-improvement systematic variants on the problems with 100 variables. The first variant stops scanning a neighbourhood once the new solution is not dominated by the archive, denoted by s-PLS$_{\not\prec}$; the second stops once it dominates an archive solution, denoted by s-PLS$_{\prec}$.

All algorithms are implemented in Java using the jMetal platform (Durillo & Nebro, 2011). All experiments are conducted on a single CPU AMD Ryzen 9 7945HX. For a given problem instance, all algorithms start from the same feasible initial solution.

Neighbourhood operators are matched to the representation of the problem (Eiben & Smith, 2015):

- Knapsack uses a 2-bit flip neighbourhood (As, by definition, 1-bit flip will increase or decrease both objective simultaneously).
- NK-landscape uses a 1-bit flip neighbourhood.
- RAP uses a one-variable integer change, i.e., increasing or decreasing one allocation by one unit.
- TSP uses the 2-opt neighbourhood.
- QAP uses the 2-swap neighbourhood.

The corresponding neighbourhood sizes are $D(D-1)/2$ for knapsack, TSP and QAP, D for NK-landscape, and $2D$ for RAP, where D denotes the problem size. If s-PLS terminates early because no unexplored archive solution remains, its final archive is carried forward until 10^6 evaluations.

We evaluate performance using the hypervolume (HV) indicator (Zitzler & Thiele, 1999). For HV computation, all objectives are converted to minimisation form. For each instance, the reference point is computed from all non-dominated solutions obtained by all algorithms up to 10^6 evaluations. Specifically, for each objective i, let $z_i^{\min}$ and $z_i^{\max}$ be the minimum and maximum values over the non-dominated solutions among the union of all the solutions obtained by the five evaluated implementations (the three main algorithms and two robustness variants) up to 10^6 evaluations. We set

$$r_i = z_i^{\max} + 0.1(z_i^{\max} - z_i^{\min}),$$

and divide the HV by the corresponding ideal-to-reference rectangle volume. This produces the normalised HV used in the main paper and in this appendix.

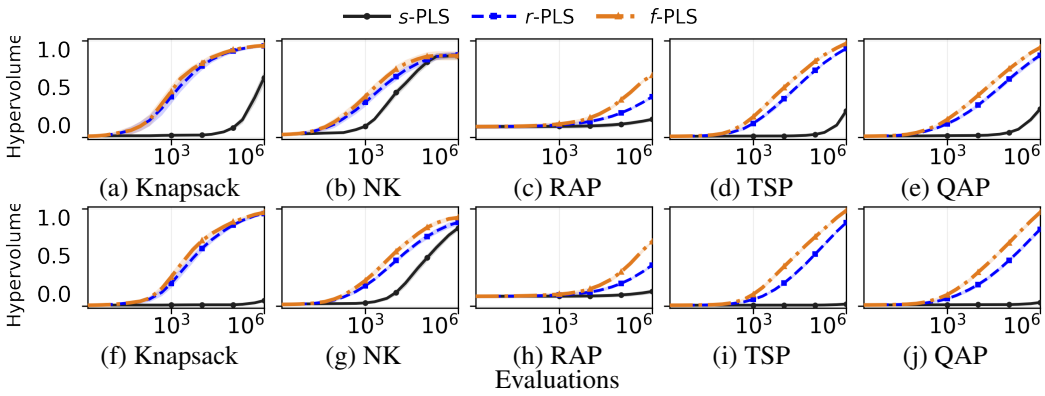


Figure A.1: Normalised HV (higher is better) of s-PLS, r-PLS, and f-PLS on the five problems with 200 and 500 variables. Curves and shaded areas show the mean and standard deviation, respectively, over 30 instances.

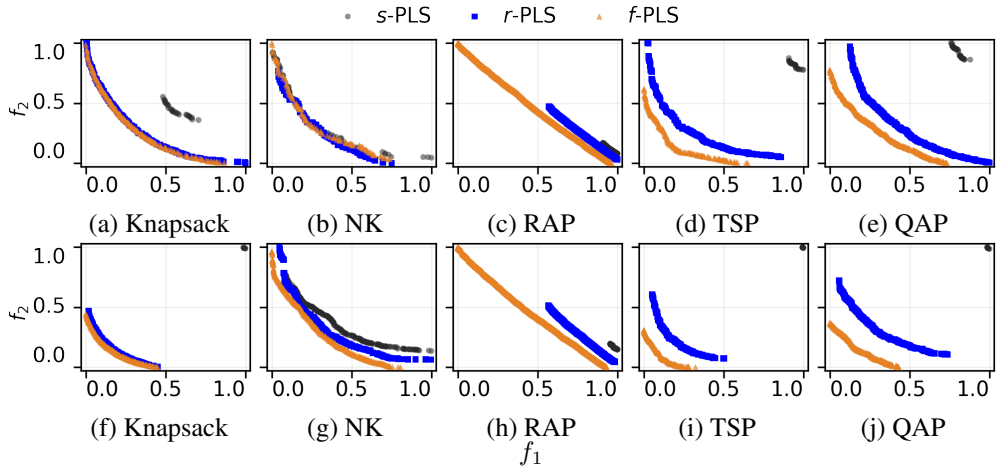


Figure A.2: All non-dominated solutions obtained by s-PLS, r-PLS, and f-PLS at 10^6 evaluations for instances with 200 variables (top row) and 500 variables (bottom row). Objectives are transformed to minimisation and scaled to $[0, 1]$; lower-left is better.

A.4 Additional Experimental Results

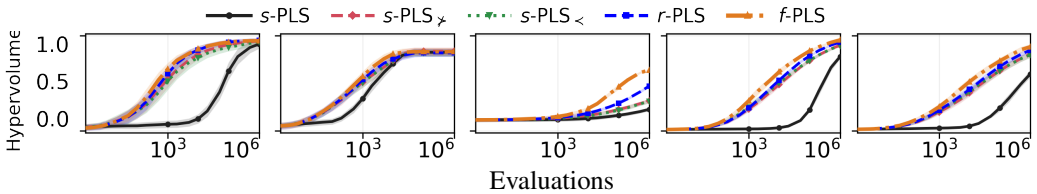
This section gives additional empirical results for problems with 200 and 500 variables. Like the main paper, Figure A.1 shows the HV trajectories and Figure A.2 gives the corresponding objective-space archives.

The main observation is consistent with the results for problems with 100 variables in the main paper. In Figure A.1, the fairer algorithms are usually faster under the reported budget. On RAP, TSP and QAP, f-PLS has a clear advantage with both 200 and 500 variables. As can be seen in Figure A.2: the f-PLS archive often reaches a wider and more advanced part of the trade-off front. Table A.1 reports the final HV results at 10^6 evaluations for all three numbers of variables. With 500 variables, the mean normalised HV of f-PLS is 0.663 on RAP, 0.980 on TSP, and 0.962 on QAP, whereas the corresponding values of r-PLS are 0.420, 0.856 and 0.788.

On Knapsack and NK-landscape, r-PLS and f-PLS have relatively similar performance throughout. On knapsack, f-PLS is still best with both 200 and 500 variables, but r-PLS is close. This suggests that randomisation already removes most of the wasted allocation caused by basic s-PLS in the early stages of the search. On NK-landscape, the difference between the algorithms is smaller with 200 variables, while f-PLS becomes clearly better with 500 variables.

Algorithm	Knapsack	NK	RAP	TSP	QAP
		100 variables			
s-PLS	0.911 (.047)	**0.825 (.041)**	0.227 (.017)	0.780 (.024)	0.596 (.037)
r-PLS	**0.951 (.021)**	**0.818 (.040)**	0.470 (.015)	0.931 (.010)	0.846 (.022)
f-PLS	**0.947 (.019)**	**0.831 (.039)**	**0.638 (.005)**	**0.955 (.007)**	**0.882 (.028)**
		200 variables			
s-PLS	0.617 (.071)	**0.841 (.034)**	0.187 (.019)	0.272 (.039)	0.293 (.032)
r-PLS	**0.948 (.015)**	**0.853 (.031)**	0.420 (.014)	0.918 (.011)	0.849 (.025)
f-PLS	**0.950 (.014)**	**0.845 (.047)**	**0.642 (.008)**	**0.971 (.004)**	**0.928 (.010)**
		500 variables			
s-PLS	0.057 (.010)	0.801 (.030)	0.151 (.021)	0.021 (.002)	0.038 (.006)
r-PLS	0.952 (.014)	0.859 (.031)	0.420 (.014)	0.856 (.013)	0.788 (.024)
f-PLS	**0.961 (.009)**	**0.909 (.018)**	**0.663 (.008)**	**0.980 (.003)**	**0.962 (.006)**

Table A.1: Normalised HV, reported as mean (standard deviation), of the three main algorithms after 10^6 evaluations. Higher values are better. Bold entries are not significantly worse than the best result in the corresponding row according to paired Wilcoxon signed-rank tests with Holm correction at the 95% confidence level.



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

Figure A.3: Normalised HV (higher is better) including the two first-improvement *s*-PLS variants on the five problems with 100 variables. Curves and bands show the mean and standard deviation of normalised HV over 30 instances.

On the other hand, on NK-landscape, *s*-PLS catches up with all problem sizes. As stated in the main paper, it progresses quickly at later stages of the search, where good neighbours are scarce for all algorithms with such a rugged landscape on NK-landscape. This is the scenario in favour of the systematic scanning to find the rare remaining good neighbours. In this case, the main slowing factor may be the exhaustion of the neighbourhood. Figure A.3 also reports the comparison with first-improvement variants of *s*-PLS on problems with 100 variables. As can be seen, although these variants' HV curves are still below that of *r*-PLS and *f*-PLS in general, the difference is much smaller. On NK-landscape, in particular, the first-improvement variants progress as quickly as their fairer PLS counterparts.

B More Observations About Good-Neighbour Distributions

This section provides more details about the distribution of good neighbours, collection procedure during the experiments, and further observations. For an archive solution x, we count how many neighbours $x' \in \mathcal{N}(x)$ are good, where a good neighbour means that x' is not dominated by the current archive. The count is denoted by G throughout.

The counts are collected from the archive of the algorithms at 10^3, 10^4, 10^5 and 10^6 evaluations. For each problem setting, all 30 generated instances are used. For small neighbourhoods, all neighbours are enumerated exactly. For large neighbourhoods, at most 2000 moves are sampled for each archive solution and the good-neighbour count is estimated from the sampled fraction. This applies mainly to large knapsack, TSP and QAP neighbourhoods.

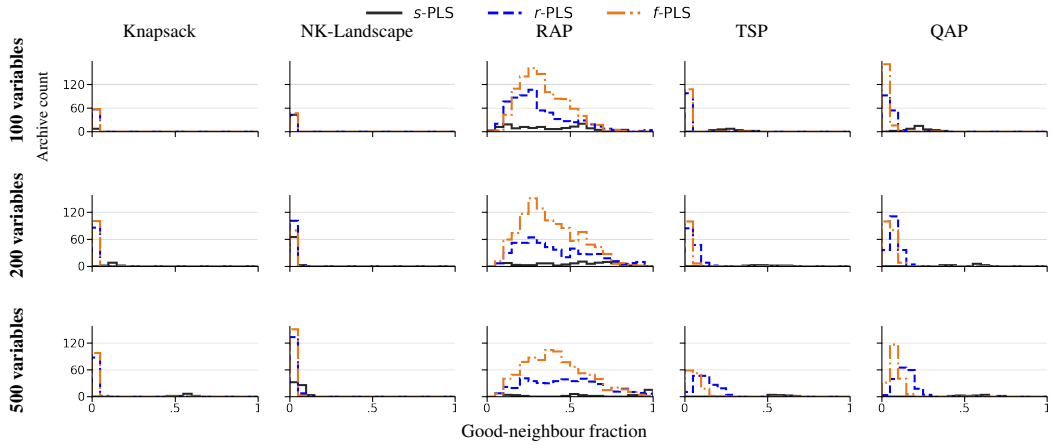


Figure B.1: Distributions of the good-neighbour fraction G/N at 10^5 evaluations in representative runs with 100 variables (top row), 200 variables (middle row), and 500 variables (bottom row). Each representative instance is selected from the 30 generated instances according to the archive-size and good-neighbour distribution profile.

Figure B.1 gives representative distributions of the good-neighbour fraction at 10^5 evaluations for problems with 100, 200 and 500 variables. As shown, for several problems, especially RAP, the archive of the basic *s*-PLS is concentrated around solutions that still have many good neighbours. This means that *s*-PLS's archive is smaller and more local: it has spent many evaluations in limited regions, so the remaining archive solutions can still look locally exploitable. The fairer methods usually have broader archives and lower good-neighbour fractions, as they have already visited more regions and pushed the archive further.

It is important to note that *s*-PLS tends to have many archive members that are still having many good neighbours. Across the five reported problems, there are 60 problem-size-checkpoint comparisons. In 52 of them, *s*-PLS alone has the highest median good-neighbour fraction across the 30 instances; in another 6, it ties for the highest. In 1599 of the 1800 paired instance comparisons, *s*-PLS has a higher within-instance median good-neighbour fraction than *r*-PLS. However, this is mainly due to the fact that *s*-PLS usually has a smaller archive. In contrast, *r*-PLS and *f*-PLS usually cover more archive solutions, and this may help their HV. This can be seen in Figure A.2, particularly on RAP, in which *r*-PLS and *f*-PLS have broader coverage.

C Proofs for the Theoretical Explanation

This section gives the formal details for the theoretical explanation section in the main paper. Proposition C.1 proves the expected waiting times for *r*-PLS and *s*-PLS. Lemma C.1 gives the two bounds on *s*-PLS. Theorems C.1 and C.2, followed by Corollaries C.1–C.4. Finally, Proposition C.2 proves the comparison between *f*-PLS and *r*-PLS.

C.1 The "Good" Neighbour Model

We analyse only the expected number of evaluations needed to find the next good neighbour (a new non-dominated neighbouring solution). The archive contains n solutions and remains fixed until the first good neighbour is found. For the comparison of *s*-PLS and *r*-PLS, both methods start from this same archive, and all neighbourhoods of *s*-PLS are initially unexplored. Systematic PLS selects uniformly among unexplored archive solutions, without replacement, and scans each selected neighbourhood in a uniformly random order. Randomised PLS selects archive solutions and their neighbours uniformly with replacement. These are fresh-state comparisons, not comparisons from partially completed scans.

Let $\mathcal{N}$ denote the neighbourhood under consideration, and let $N = |\mathcal{N}|$ denote its size. For an archive solution selected uniformly from these n solutions, let

$$G \in \{0, 1, \ldots, N\}$$

be the number of good neighbours of that solution. The distribution of G is the empirical categorical distribution over the archive; no named distributional assumption (such as the geometric distribution) is imposed. Write

$$q := \mathbb{P}(G = 0), \qquad r := 1 - q,$$
$$\mu := \mathbb{E}[G], \qquad \mu^+ := \mathbb{E}[G \mid G > 0].$$

Here nq and nr are integer counts (as the number of solutions is discrete). When $q < 1$, we have $\mu = r\mu^+$. When $q = 1$, no selected archive solution has any good neighbour, and all algorithms stagnate in this fixed state.

We first establish a frequently used auxiliary lemma for systematic scanning in s-PLS.

Auxiliary Lemma C.1 (Position of the first marked element)**.** *In a uniformly random ordering of L elements, of which k are marked, where $1 \leq k \leq L$, the position J of the first marked element satisfies*

$$\mathbb{E}[J] = \frac{L+1}{k+1}.$$

Proof. The k marked elements divide the $L - k$ unmarked elements into $k + 1$ gaps, including the gaps before the first and after the last marked element. By symmetry, each gap contains $(L-k)/(k+1)$ elements in expectation. The first marked position is one plus the first gap size, giving

$$\mathbb{E}[J] = 1 + \frac{L-k}{k+1} = \frac{L+1}{k+1}.$$

This is the standard order-statistic argument for sampling without replacement (Knuth, 1997, Sec. 3.4.1). Taking $L = N$ and marking the good neighbours yields the expected first-good-neighbour position $(N+1)/(k+1)$. □

C.2 Expected Times of r-PLS and s-PLS

Proposition C.1 (Expected number of evaluations to find the next good neighbour)**.** *Assume $0 \leq q < 1$ and $\mu > 0$. Let $T_{r\text{-PLS}}$ and $T_{s\text{-PLS}}$ be the numbers of evaluations until the next good neighbour is found. Under the fixed-state model,*

$$\mathbb{E}[T_{r\text{-PLS}}] = \frac{N}{\mu}, \tag{C.1}$$

$$\mathbb{E}[T_{s\text{-PLS}}] = \left(\frac{n+1}{nr+1} - 1\right) N + (N+1)\mathbb{E}\left[\frac{1}{G+1} \mid G > 0\right]. \tag{C.2}$$

Proof. ***r*-PLS.** In one evaluation, r-PLS first selects an archive solution, and then samples one neighbour uniformly at random. If the selected solution has G good neighbours, then the success probability of this evaluation is G/N. Averaging over the archive state gives

$$p_{\text{succ}} = \mathbb{E}\left[\frac{G}{N}\right] = \frac{\mathbb{E}[G]}{N} = \frac{\mu}{N}.$$

The number of evaluations until the first success is geometric with mean $1/p_{\text{succ}}$. Therefore

$$\mathbb{E}[T_{r\text{-PLS}}] = \frac{1}{p_{\text{succ}}} = \frac{N}{\mu}.$$

***s*-PLS.** The search process of s-PLS has two parts.

For the first part, uniform selection among unexplored archive solutions induces a uniformly random ordering of the n archive members. Exactly nr members are promising. By Auxiliary Lemma C.1,

the expected position of the first promising member is $(n+1)/(nr+1)$. Subtracting one counts the non-promising members preceding it. Each such member costs N evaluations, so their expected total cost is

$$\left(\frac{n+1}{nr+1} - 1\right) N = \frac{nqN}{nr+1}.$$

For the second part, once a solution with $G > 0$ is selected, Auxiliary Lemma C.1 gives the expected scan length. The first promising member is uniform among the nr promising members. Conditioning on its good-neighbour count G,

$$\mathbb{E}[J \mid G] = \frac{N+1}{G+1}.$$

Taking the expectation over the conditional distribution of G given $G > 0$ gives

$$\mathbb{E}[J \mid G > 0] = (N+1)\mathbb{E}\left[\frac{1}{G+1} \mid G > 0\right].$$

Adding the two parts gives the result. □

C.3 Bounds for s-PLS

Lemma C.1 (Bounds for systematic PLS)**.** *Assume* $0 \le q < 1$. *Then*

$$\frac{nqN}{nr+1} + \frac{N+1}{\mu^+ + 1} \le \mathbb{E}[T_{s\text{-}PLS}],$$

$$\mathbb{E}[T_{s\text{-}PLS}] \le \frac{nqN}{nr+1} + 1 + \frac{N-\mu^+}{2}.$$

Proof. Let

$$h(x) = \frac{1}{x+1}.$$

This function is convex on $x \ge 0$.

Lower bound. By Jensen's inequality (Boyd & Vandenberghe, 2004), for a convex function,

$$\mathbb{E}[h(G) \mid G > 0] \ge h(\mathbb{E}[G \mid G > 0]) = \frac{1}{\mu^+ + 1}.$$

Multiplying by $N+1$ and adding the no-good-neighbour term gives

$$\mathbb{E}[T_{s\text{-PLS}}] \ge \frac{nqN}{nr+1} + \frac{N+1}{\mu^+ + 1}.$$

Upper bound. If $N = 1$, every positive count equals one, and the bound holds with equality. Hence assume $N \ge 2$ for the chord argument. Conditional on $G > 0$, the value of G lies in $\{1, \dots, N\}$. For a convex function, its graph lies below the chord joining the two endpoints on an interval. Thus, for every $x \in [1, N]$,

$$h(x) \le \frac{N-x}{N-1} h(1) + \frac{x-1}{N-1} h(N).$$

Substituting $h(1) = 1/2$ and $h(N) = 1/(N+1)$, and taking the expectation conditional on $G > 0$, we get

$$\mathbb{E}[h(G) \mid G > 0] \le \frac{N-\mu^+}{N-1} \cdot \frac{1}{2} + \frac{\mu^+ - 1}{N-1} \cdot \frac{1}{N+1}.$$

Multiplying by $N+1$,

$$(N+1)\mathbb{E}[h(G) \mid G > 0] \le \frac{(N+1)(N-\mu^+)}{2(N-1)} + \frac{\mu^+ - 1}{N-1}$$

$$= 1 + \frac{N-\mu^+}{2}.$$

Adding the no-good-neighbour term gives

$$\mathbb{E}[T_{s\text{-PLS}}] \le \frac{nqN}{nr+1} + 1 + \frac{N-\mu^+}{2}.$$

□

C.4 When is r-PLS Faster?

For compactness, we write

$$A := \frac{nqN}{nr+1}, \qquad C := r - \frac{Nq}{nr+1}.$$

Here A is the expected cost of scanning non-promising archive members before the first promising one.

Theorem C.1 (When randomised PLS is faster)**.** *Assume $0 < q < 1$ and $\mu > 0$. If*

$$\mu \geq \mu_{\text{high}}(q,n) := \frac{-C + \sqrt{C^2 + 4ANr}}{2A},$$

then $\mathbb{E}[T_{r\text{-PLS}}] \leq \mathbb{E}[T_{s\text{-PLS}}]$.

Proof. By Proposition C.1 and Lemma C.1, it suffices that

$$\frac{N}{\mu} \leq A + \frac{N+1}{\mu^+ + 1} = A + \frac{(N+1)r}{\mu + r}.$$

Multiplying by the positive quantity $\mu(\mu + r)$ and rearranging gives

$$A\mu^2 + \big(Ar + (N+1)r - N\big)\mu - Nr \geq 0.$$

Since $q + r = 1$, the linear coefficient is

$$\begin{aligned} Ar + (N+1)r - N &= Ar + r - Nq \\ &= r - \frac{Nq}{nr+1} = C. \end{aligned}$$

Thus, it suffices that $A\mu^2 + C\mu - Nr \geq 0$. As $A > 0$ and $Nr > 0$, this quadratic has one negative and one positive root. Its positive root is $\mu_{\text{high}}(q,n)$, so the inequality holds for every μ at least this root. □

C.5 When is s-PLS Faster?

Theorem C.2 (When systematic PLS is faster)**.** *Assume $N \geq 2$, $0 \leq q < 1$ and $\mu > 0$, and let*

$$B := 2rA + r(N+2).$$

If

$$\mu \leq \mu_{\text{low}}(q,n) := \frac{B - \sqrt{B^2 - 8Nr}}{2},$$

then $\mathbb{E}[T_{s\text{-PLS}}] \leq \mathbb{E}[T_{r\text{-PLS}}]$.

Proof. By Proposition C.1 and Lemma C.1, it suffices that

$$A + 1 + \frac{N - \mu/r}{2} \leq \frac{N}{\mu}.$$

Multiplication by $2r\mu > 0$ and rearrangement give

$$\mu^2 - \big(2rA + r(N+2)\big)\mu + 2Nr \geq 0,$$

or $\mu^2 - B\mu + 2Nr \geq 0$. To verify that the roots are real, let $m = nr \geq 1$. Since $N \geq 2$,

$$A = \frac{Nnq}{m+1} \geq \frac{nq}{m} = \frac{q}{r}.$$

The quadratic is positive at zero, while at $\mu = rN$ its value is

$$2rN\big(1 - r(A+1)\big) \leq 0.$$

It therefore has real positive roots (possibly coincident). It is non-negative to the left of its smaller root, which is $\mu_{\text{low}}(q,n)$. This establishes the sufficient condition. □

C.6 Other Simple Cases

Corollary C.1 (All neighbourhoods are identical)**.** *If $G = g$ where $1 \leq g \leq N$, then* s*-PLS is no slower than* r*-PLS. It is faster except for the trivial case $g = N$, where they are equal.*

Proof. Here $q = 0$ and $\mu = g$. For *r*-PLS,

$$\mathbb{E}[T_{r\text{-PLS}}] = \frac{N}{g}.$$

For *s*-PLS,

$$\mathbb{E}[T_{s\text{-PLS}}] = \frac{N+1}{g+1}.$$

Compare the two:

$$\frac{N+1}{g+1} \leq \frac{N}{g}$$
$$gN + g \leq gN + N,$$

which is true because $g \leq N$. It is usually strict, as in general $g < N$ (i.e., not all neighbours are good). □

Corollary C.2 (All promising and small mean)**.** *If $q = 0$ and $\mu \leq 2$, then* s*-PLS is no slower than* r*-PLS for any distribution of G with this mean.*

Proof. When $q = 0$, we have $r = 1$ and $\mu^+ = \mu$. The upper bound of *s*-PLS is

$$\mathbb{E}[T_{s\text{-PLS}}] \leq 1 + \frac{N-\mu}{2}.$$

We want this upper bound to be at most $\mathbb{E}[T_{r\text{-PLS}}] = N/\mu$. That is,

$$1 + \frac{N-\mu}{2} \leq \frac{N}{\mu}.$$

For $N \geq 2$, Theorem C.2 has $A = 0$, $B = N + 2$ and smaller root 2 when $q = 0$. For $N = 1$, all positive counts equal one and both methods succeed immediately. Therefore, even the worst case for *s*-PLS is no worse than *r*-PLS. □

Corollary C.3 (All promising and large mean)**.** *For $q = 0$ and $2 < \mu < N$, then either algorithm can be faster. Homogeneous distributions favour* s*-PLS. Highly heterogeneous distributions can favour* r*-PLS. If $\mu = N$, then all neighbours are good and the two algorithms are equal.*

Proof. Again $q = 0$ and $\mu^+ = \mu$. The lower bound of *s*-PLS is

$$\frac{N+1}{\mu+1}.$$

If μ is an integer, the homogeneous distribution $G = \mu$ attains this lower bound. For $\mu < N$,

$$\frac{N+1}{\mu+1} < \frac{N}{\mu} = \mathbb{E}[T_{r\text{-PLS}}],$$

so this distribution favours *s*-PLS. If μ is not an integer, the analogous distribution supported on the two adjacent integers $\lfloor \mu \rfloor$ and $\lceil \mu \rceil$ also gives $\mathbb{E}[T_{s\text{-PLS}}] < N/\mu$ by direct substitution.

On the other hand, the upper bound of *s*-PLS is

$$1 + \frac{N-\mu}{2}.$$

When $\mu > 2$,

$$\frac{N}{\mu} < 1 + \frac{N-\mu}{2}.$$

This upper bound is attained by the endpoint distribution

$$\mathbb{P}(G = 1) = \frac{N - \mu}{N - 1}, \qquad \mathbb{P}(G = N) = \frac{\mu - 1}{N - 1}.$$

Thus, this highly heterogeneous distribution makes *r*-PLS faster.

For the trivial case $\mu = N$ (all neighbours are good), then G is always equal to N. In this case, both algorithms then find a good neighbour in exactly one evaluation. □

Corollary C.4 (Stagnation)**.** *If $q = 1$, then G always equal to 0. In this fixed state, all three algorithms need infinite expected time to find the next good neighbour.*

Proof. If $G = 0$ for every archive solution, no evaluation can find a good neighbour. So the time to success is infinite for *s*-PLS, *r*-PLS, and *f*-PLS. □

C.7 *r*-PLS VERSUS THE EVEN FAIRER *f*-PLS

Proposition C.2 (Fair PLS is no slower than randomised PLS)**.** *Consider a fixed archive with ℓ solutions. Let p_i be the probability that one random neighbour of archive solution i is good. Assume* f*-PLS breaks ties uniformly at random among the least-selected archive solutions. For every integer $t \geq 0$,*

$$\mathbb{P}(T_{f\text{-}PLS} > t) \leq \mathbb{P}(T_{r\text{-}PLS} > t).$$

Consequently,

$$\mathbb{E}[T_{f\text{-}PLS}] \leq \mathbb{E}[T_{r\text{-}PLS}].$$

The inequality is strict when the p_i values are not all equal.

Proof. Let

$$\bar{p} = \frac{1}{\ell} \sum_{i=1}^{\ell} p_i, \qquad y_i = 1 - p_i.$$

where y_i is the failure probability for one evaluation of solution i (i.e., fail to find a good neighbour of solution i).

The repeated-selection process of *r*-PLS. Since *r*-PLS samples archive solutions with replacement, and each evaluation succeeds with probability $\bar{p}$, for every integer $t \geq 0$,

$$\mathbb{P}(T_{r\text{-PLS}} > t) = (1 - \bar{p})^t.$$

The fair process of *f*-PLS. Since *f*-PLS samples each archive solution once before sampling any solution a second time, the search can be viewed in rounds of length ℓ. Write

$$t = a\ell + b, \qquad a \geq 0,\ 0 \leq b < \ell.$$

After a complete failed rounds, every solution has failed a times. The probability of this is

$$\left(\prod_{i=1}^{\ell} y_i \right)^a.$$

In the next partial round, *f*-PLS samples b distinct archive solutions. Since tie-breaking is uniform, this is a uniformly random subset of size b. The average probability that all these b sampled solutions fail is

$$\frac{e_b(y_1, \ldots, y_\ell)}{\binom{\ell}{b}},$$

where e_b is the elementary symmetric sum of degree b, i.e., the sum of all products of b distinct y_i values. Therefore,

$$\mathbb{P}(T_{f\text{-PLS}} > t) = \left(\prod_{i=1}^{\ell} y_i \right)^a \frac{e_b(y_1, \ldots, y_\ell)}{\binom{\ell}{b}}.$$

Using Maclaurin's inequality (Hardy et al., 1988), for non-negative numbers $y_1, \ldots, y_m$,

$$\left(\frac{e_b(y_1, \ldots, y_\ell)}{\binom{\ell}{b}} \right)^{1/b} \leq \frac{e_1(y_1, \ldots, y_\ell)}{\ell}$$

for every $b = 1, \ldots, \ell$. In words, the average product of b distinct values is no larger than the b-th power of their average. Since

$$\frac{e_1(y_1, \ldots, y_\ell)}{\ell} = \frac{1}{\ell} \sum_{i=1}^{\ell} (1 - p_i) = 1 - \bar{p},$$

we have

$$\frac{e_b(y_1, \ldots, y_\ell)}{\binom{\ell}{b}} \leq (1 - \bar{p})^b.$$

For $b = \ell$, the same inequality gives

$$\prod_{i=1}^{\ell} y_i \leq (1 - \bar{p})^\ell.$$

Plugging in the *r*-PLS and f-PLS, we have

$$\begin{aligned} &\mathbb{P}(T_{f\text{-PLS}} > t) = \\ &\left(\prod_{i=1}^{\ell} y_i \right)^a \frac{e_b(y_1, \ldots, y_\ell)}{\binom{\ell}{b}} \leq \left((1 - \bar{p})^\ell \right)^a (1 - \bar{p})^b \\ &\qquad = (1 - \bar{p})^{a\ell + b} \\ &\qquad = (1 - \bar{p})^t \\ &\qquad = \mathbb{P}(T_{r\text{-PLS}} > t). \end{aligned}$$

Since the tail probability of f-PLS is no larger at every time t, with $\mathbb{E}[T] = \sum_{t \geq 0} \mathbb{P}(T > t)$, we have

$$\mathbb{E}[T_{f\text{-PLS}}] \leq \mathbb{E}[T_{r\text{-PLS}}].$$

If the p_i values are not all equal, Maclaurin's inequality is strict for at least one round length, so the inequality is strict. □

C.8 Interpretation

The analysis gives the following picture. If good neighbours are evenly distributed across archive solutions, systematic scanning can be very efficient, as it never visit the same neighbour in a neighbourhood. If some archive solutions have no good neighbours, and if the remaining promising solutions are uneven, then systematic scanning can waste many evaluations a few several solutions. Randomisation reduces this waste. Fair allocation reduces it further, as it avoids selecting the same archive solution too many times before other archive solutions are touched.

D Proofs of the Two Toy Examples

In the main paper, we use two toy examples to show why fairness can help and why it does not always help. Both examples contain four archive solutions and a common neighbourhood size N. The question is how many evaluations are needed, in expectation, to find the next good neighbour.

D.1 Example 1: One Good Neighbourhood and Three Bad Neighbourhoods

Proposition D.1 (Fairness helps in Example 1)**.** *Suppose the archive contains four solutions. Three solutions have no good neighbours, and one solution has all neighbours good. Then*

$$\mathbb{E}[T_{f\text{-PLS}}] = 2.5, \quad \mathbb{E}[T_{r\text{-PLS}}] = 4, \quad \mathbb{E}[T_{s\text{-PLS}}] = 1.5N + 1.$$

Thus f*-PLS is faster than* r*-PLS, and* r*-PLS is faster than* s*-PLS when $N > 2$. When $N = 2$,* r*-PLS and* s*-PLS tie in expectation.*

Proof. For *r*-PLS, one archive solution is selected uniformly at random. Only one of the four archive solutions is promising. If this solution is selected, every neighbour is good. Therefore the success probability of one evaluation is

$$p_{\text{succ}} = \frac{1}{4},$$

and

$$\mathbb{E}[T_{r\text{-PLS}}] = \frac{1}{p_{\text{succ}}} = 4.$$

For f-PLS, the four archive solutions are selected once each in a fair round. The promising solution is equally likely to appear in position 1, 2, 3 or 4 of this round. Once it is reached, the first neighbour is good. Thus

$$\mathbb{E}[T_{f\text{-PLS}}] = \frac{1+2+3+4}{4} = 2.5.$$

For *s*-PLS, the algorithm may first scan non-promising solutions. A non-promising solution costs N evaluations because its whole neighbourhood is exhausted. The position of the only promising solution in a random order of four archive solutions has expectation

$$\frac{1+2+3+4}{4} = 2.5.$$

So the expected number of non-promising solutions before the promising one is $2.5 - 1 = 1.5$. When the promising solution is reached, only one evaluation is needed because all its neighbours are good. Therefore

$$\mathbb{E}[T_{s\text{-PLS}}] = 1.5N + 1.$$

Comparing the three values gives the claim. □

The key reason is that *s*-PLS can spend N evaluations scanning the neighbourhood of a non-promising solution. *r*-PLS avoids spending all N evaluations on that solution. f-PLS goes one step further: it also avoids selecting the same archive solution repeatedly before the other archive solutions are touched.

D.2 Example 2: All Neighbourhoods are Equally Promising

Proposition D.2 (Systematic scanning helps in Example 2)**.** *Suppose the archive contains four solutions. Every archive solution has exactly $N/4$ good neighbours. Assume N is divisible by 4. Then*

$$\mathbb{E}[T_{r\text{-PLS}}] = \mathbb{E}[T_{f\text{-PLS}}] = 4,$$

while

$$\mathbb{E}[T_{s\text{-PLS}}] = \frac{N+1}{N/4+1} < 4.$$

Thus s*-PLS is faster in expectation.*

Proof. For *r*-PLS, whichever archive solution is selected, a uniformly sampled neighbour is good with probability

$$\frac{N/4}{N} = \frac{1}{4}.$$

Thus

$$\mathbb{E}[T_{r\text{-PLS}}] = 4.$$

For f-PLS, the order of archive-solution selection no longer matters, as every archive solution has the same success probability $1/4$. Each evaluation therefore succeeds with probability $1/4$, and

$$\mathbb{E}[T_{f\text{-PLS}}] = 4.$$

For *s*-PLS, the selected archive solution has $N/4$ good neighbours. By Auxiliary Lemma C.1, the expected position of the first good neighbour in a random scan is

$$\mathbb{E}[T_{s\text{-PLS}}] = \frac{N+1}{N/4+1}.$$

Finally,

$$\frac{N+1}{N/4+1} < 4$$

because $N + 1 < N + 4$. Therefore, *s*-PLS is faster in this balanced case. □

This example explains why fairness is not an unconditional advantage. When all archive solutions are similarly promising, changing the order of archive-solution selection gives little benefit. In this case, the main advantage comes from scanning without replacement (i.e., not visiting the same neighbour) inside the selected neighbourhood.