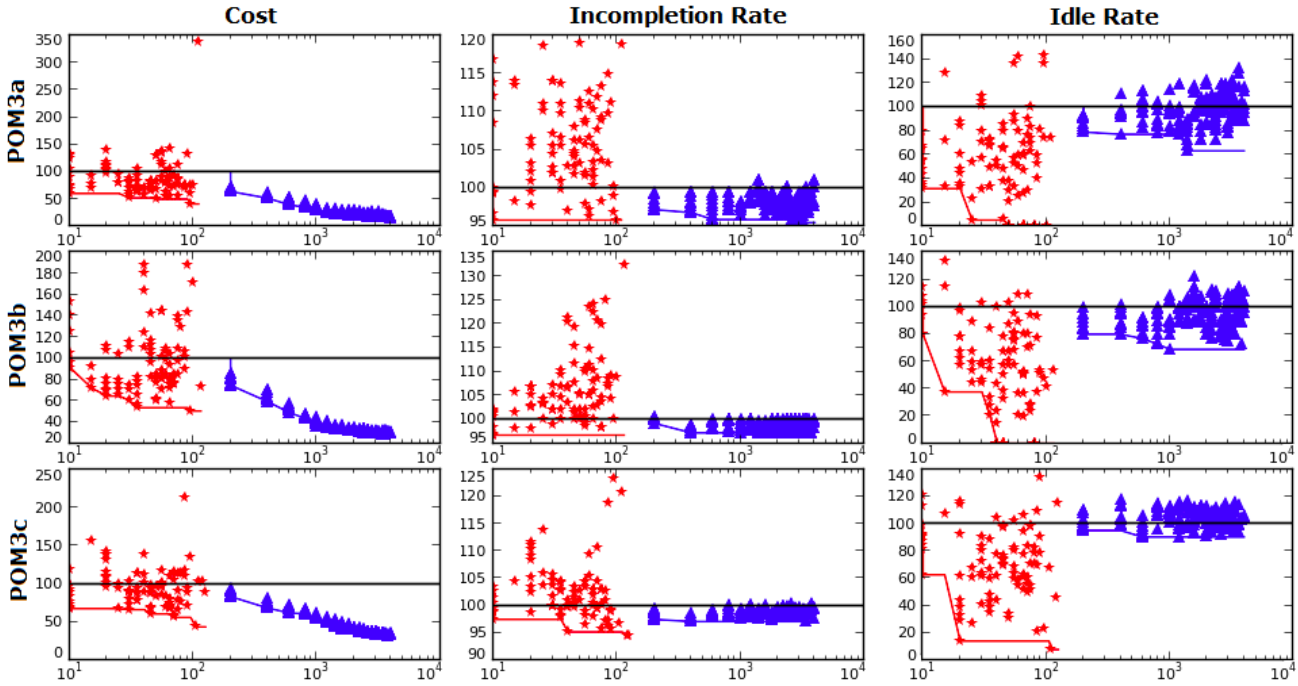

Fig. 17. POM results: 20 repeats of each MOEA (one row per scenario) from **GALE (red)** and **NSGA-II (blue)**. Each y-axis represents the percent objective value relative to that in the initial baseline population, and lower is better. The lines trend across the best (lowest) seen objective thus far. Each x-axis shows number of evaluations (log scale.

Figure 16 are the values that are statistically best (this test was applied across all the values in each row using a Mann-Whitney 99% confidence procedure). Measured in terms of number of wins (the gray cells), GALE wins more than the other MOEAs. Also, when it is not the best in each row, it loses by very small amounts (never more than 3%).

An issue with summary statistics like Figure 16 is that such summaries can hide significant effects. For example, all the Figure 16 results from POM3 are very close to 90% suggesting that, for this model, all MOEAs produced similar results. Yet a closer inspection of those results shows that this is not the case. Figure 17 shows how NSGA-II and GALE evolved candidates with better objective scores. The y-vertical-axis denotes changes from the median of the initial population. That is:

- $Y = 50$ would indicate that we have halved the value of some objective;
- $Y > 100$ (over the solid black line) would indicate that optimization failed to improve this objective.

In Figure 17, all the y-axis values are computed such that *lower* values are *better*. Specifically, the results in the column labeled *Incompletion Rate* is the ratio *initial/now* values. Hence, if we are *now* completing a larger percentage of the requirements, then *incompletion* is better if it is less than 100%; i.e.

$$Incompletion\% = 100 - Completion\%$$

At first glance, these results seem to say that GALE performed worse than NSGA-II since NSGA-II achieved larger *Cost* reductions. However, the *Idle* results show otherwise: NSGA-II rarely reduced the *Idle* time of the developers while GALE found ways to achieve reductions down to bear zero percent *Idle*.

This observation begs the question: how could NSGA-II reduce cost while increasing developer *Idle* time? One answer is some quirk in the input space. However, that answer is not supported; observe that the same strange pattern (of increased *Idle* and decreased *Cost*) holds in POM3a and POM3b and POM3c).

A better explanation can be found in the *Incompletion Rate* results: NSGA-II told the developers to complete fewer requirements. Since POM3 measures cost in terms of the *salary times effort* for the completed requirements, then by completing fewer requirements, NSGA-II could reduce the reported cost. Note that this is not necessarily an error in the POM3 costing routines- providing that an optimizer also avoids leaving programmers idle. In this regard, NSGA-II is far worse than GALE since the latter successfully reduces *cost* as well as the *Idle Rate*.

We conjecture that NSGA-II's failure in this regard was due to its use of a binary dominance function. As discussed in §2.3, continuous domination functions, such as that used by GALE, can find more nuances in the trade-offs between competing objectives. In support of this conjecture, we note that SPEA2, which also uses binary domination, also suffers from large *Idle Rates* in the final frontier of POM3 outputs[2].

2. Those SPEA2 results look very similar to the NSGA-II results of Figure 17. Those results are not shown here due to space reasons.