Supplementary material

for paper “Multi-Metric Algorithmic Complexity: Beyond Asymptotic Analysis”, Kavun Sergii

Table 1. Comprehensive summary report

| Metric | Value | | Metric | Value |
|---|---|---|---|---|
| Analysis Timestamp | Linux-6.1.123+-x86_64-with-glibc2.35 | | Best Algo (Constant_O(1)_Formula) - CU | 17 |
| Architecture | x86_64 | | Best Algo (Constant_O(1)_Formula) - EU | 0.00148 |
| Profile - Profile | RESEARCH | | Best Algo (Constant_O(1)_Formula) - CO2 | 0.000498 |
| Profile - Weights | {'CU': 0.4, 'EU': 0.3, 'CO2': 0.25, '$': 0.05} | | Best Algo (Constant_O(1)_Formula) - $ | 0.000169 |
| Profile - Description | Research/Academic - focused on performance wit... | | Best Algo (Constant_O(1)_Formula) - CU_normalized | 100 |
| Stat - Best Algorithm | Constant_O(1)_Formula | | Best Algo (Constant_O(1)_Formula) - EU_normalized | 100 |
| Stat - Worst Algorithm | Sqrt_O(sqrt_n)_PrimalityTest | | Best Algo (Constant_O(1)_Formula) - CO2_normalized | 100 |
| Stat - Average Composite Score | 64.85 | | Best Algo (Constant_O(1)_Formula) - $_normalized | 100 |
| Stat - Median Composite Score | 77.76 | | Best Algo (Constant_O(1)_Formula) - COMPOSITE_SCORE | 100 |
| Stat - Composite Score Std | 30.05 | | Best Algo (Constant_O(1)_Formula) - SCORE_GRADE | A+ |

| Metric | Value | Metric | Value |
|---|---|---|---|
| Stat - Score Range | 100.00 | Best Algo (Constant_O(1)_Formula) - EFFICIENCY_RATING | Excellent |
| Recommendation #1 | Analysis performed using 'RESEARCH' profile: R... | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - CU | 129 |
| Recommendation #2 | Use 'Constant_O(1)_Formula' for optimal perfor... | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - EU | 0.01115 |
| Recommendation #3 | Avoid 'Sqrt_O(sqrt_n)_PrimalityTest' due to po... | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - CO2 | 0.003637 |
| Recommendation #4 | For energy-critical applications, consider 'Sq... | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - $ | 0.001285 |
| Recommendation #5 | For environmentally conscious deployment, 'Sqr... | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - CU_normalized | 0 |
| Recommendation #6 | Large performance variation detected - conside... | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - EU_normalized | 0 |
| Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - CO2_normalized | 0 | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - $_normalized | 0 |
| Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - COMPOSITE_SCORE | 0 | Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - EFFICIENCY_RATING | Poor |

| Metric | Value | Metric | Value |
|---|---|---|---|
| Worst Algo (Sqrt_O(sqrt_n)_PrimalityTest) - SCORE_GRADE | F | | |

Baseline for comparison is 'Constant_O(1)_Formula'.

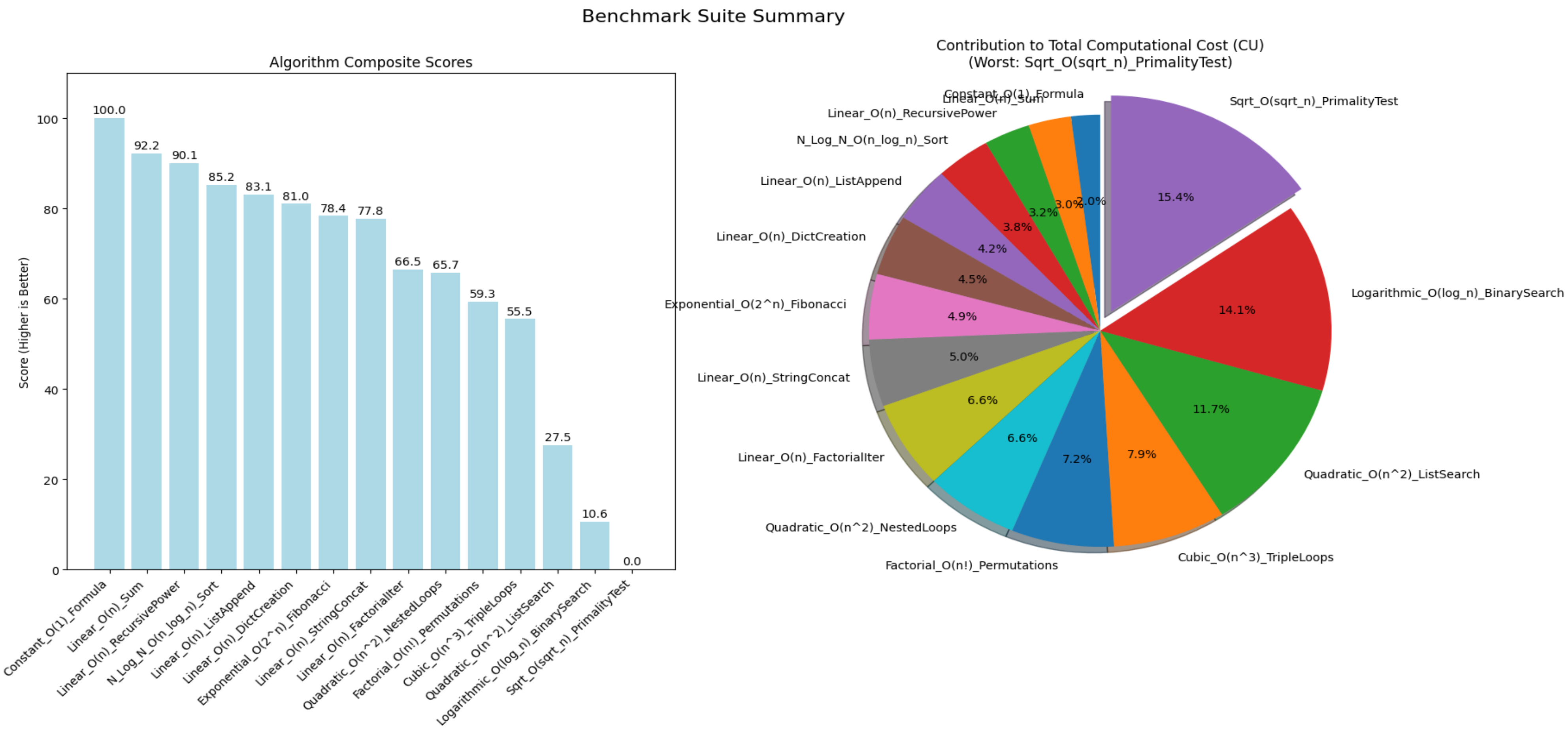


Fig. 1. Generating overall benchmark summary chart

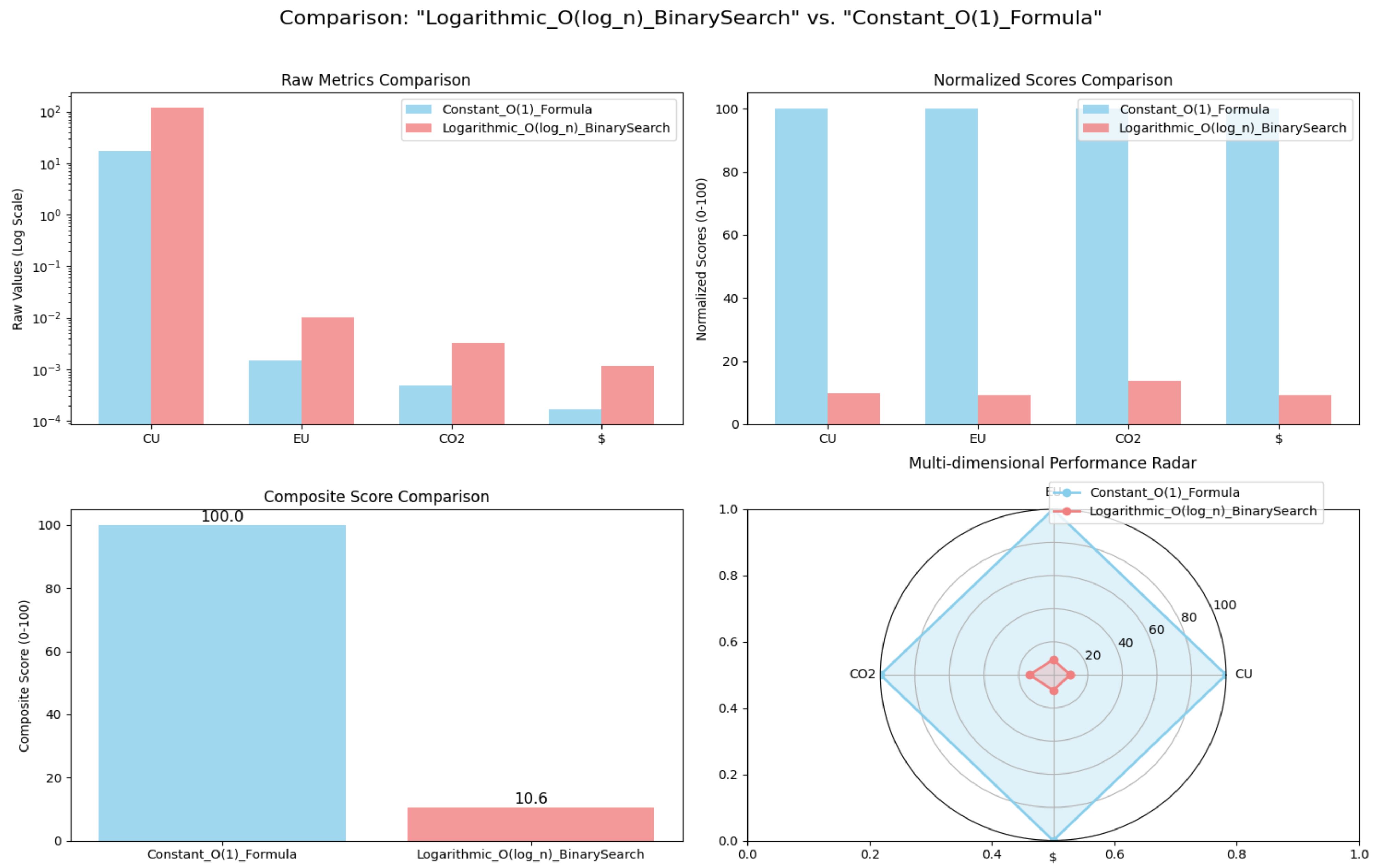


Fig. 2. Generating chart: comparison_Logarithmic_Olog_n_BinarySearch_vs_Constant_O1_Formula

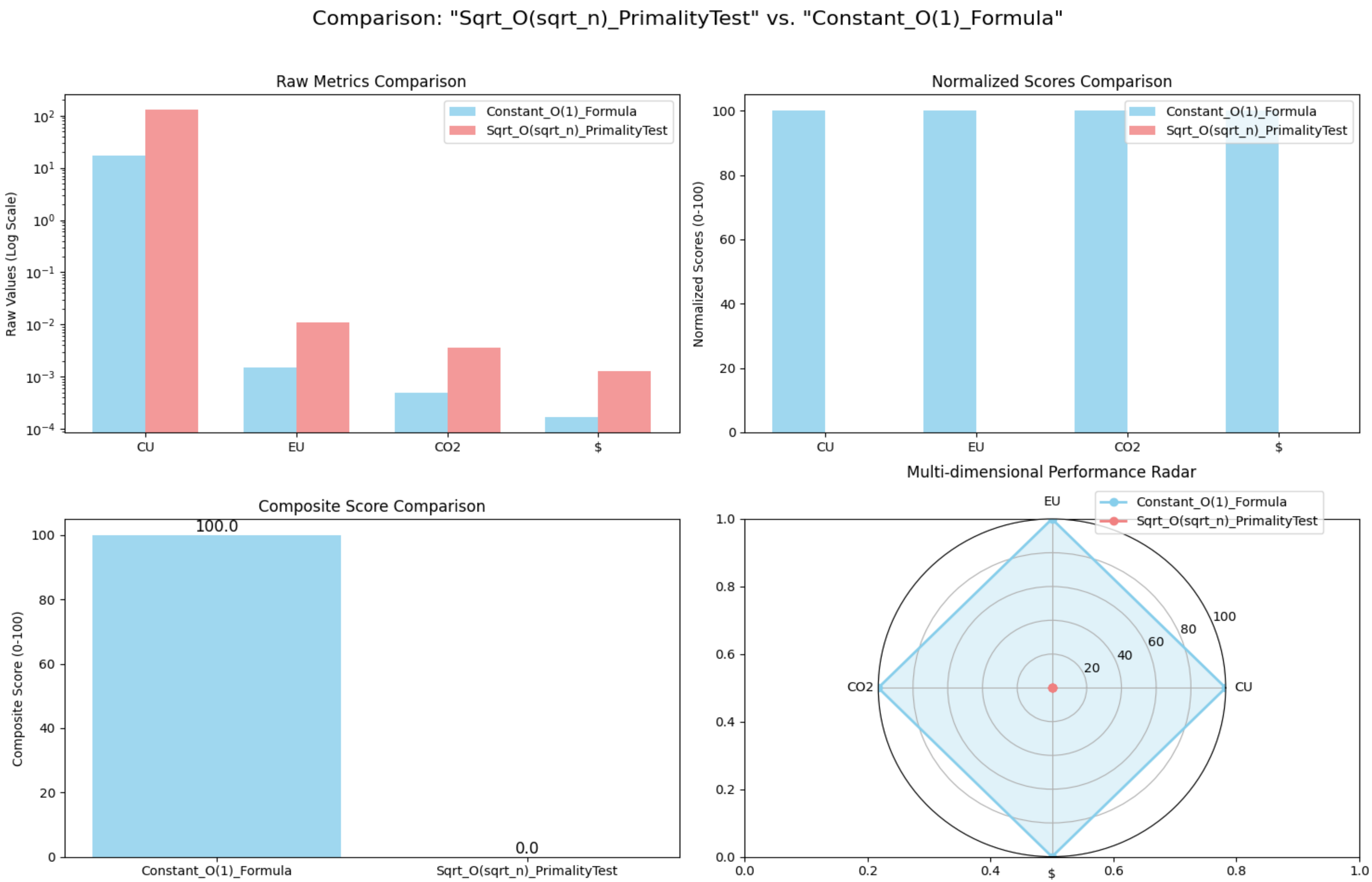


Fig. 3. Generating chart: comparison_Sqrt_Osqrt_n_PrimalityTest_vs_Constant_O1_Formula

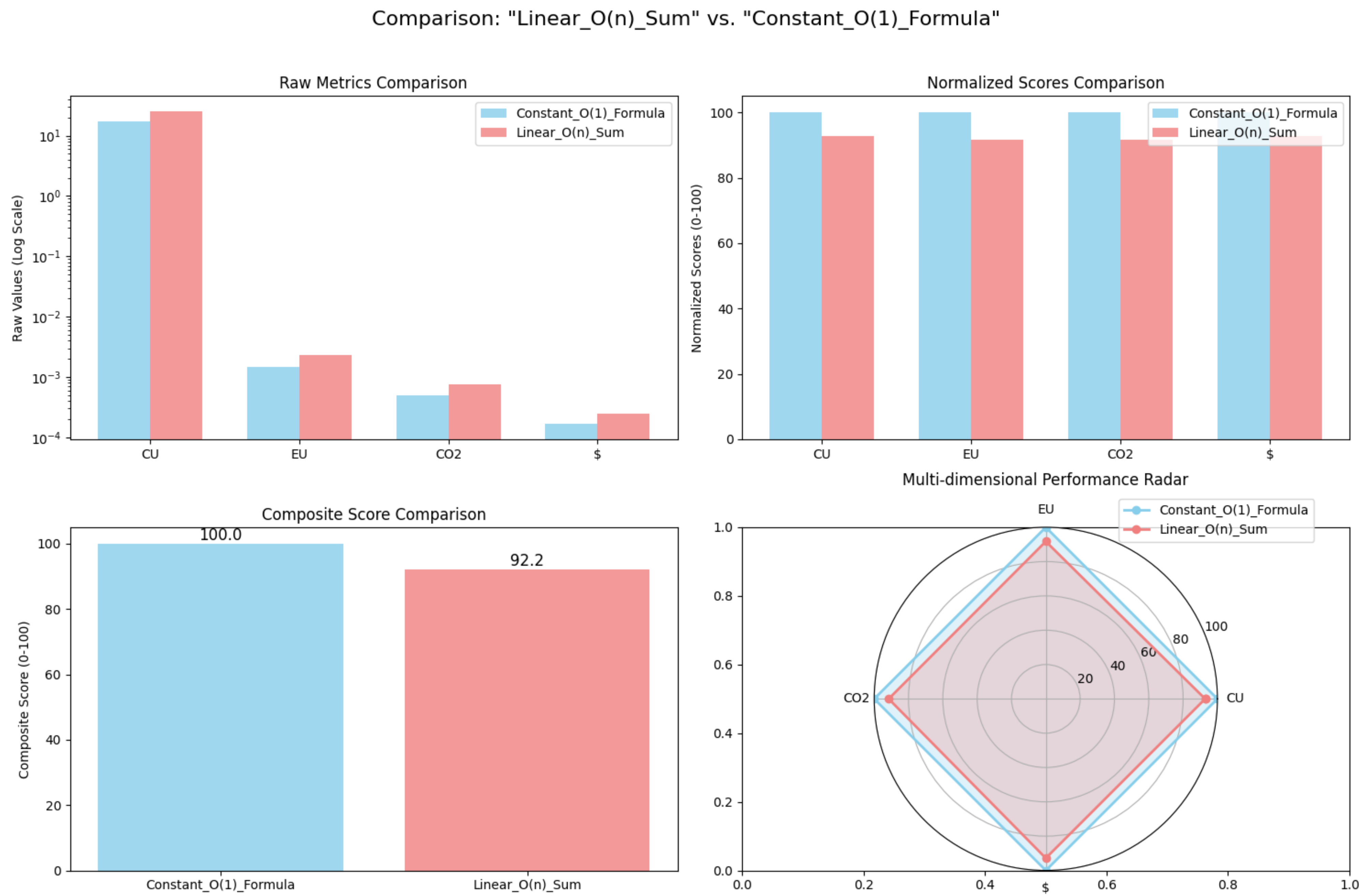


Fig. 4. Generating chart: comparison_Linear_On_Sum_vs_Constant_O1_Formula

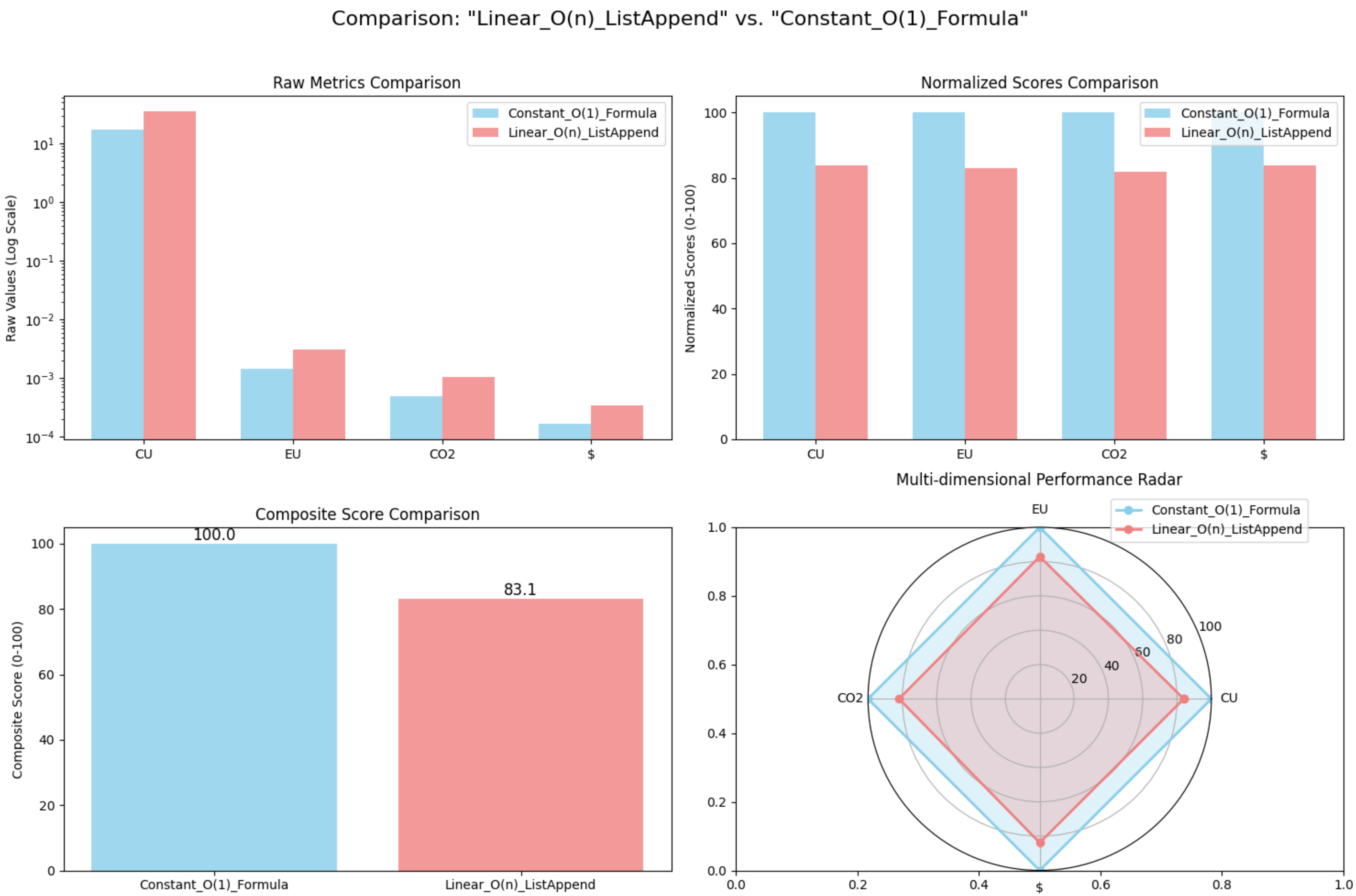


Fig. 5. Generating chart: comparison_Linear_On_ListAppend_vs_Constant_O1_Formula

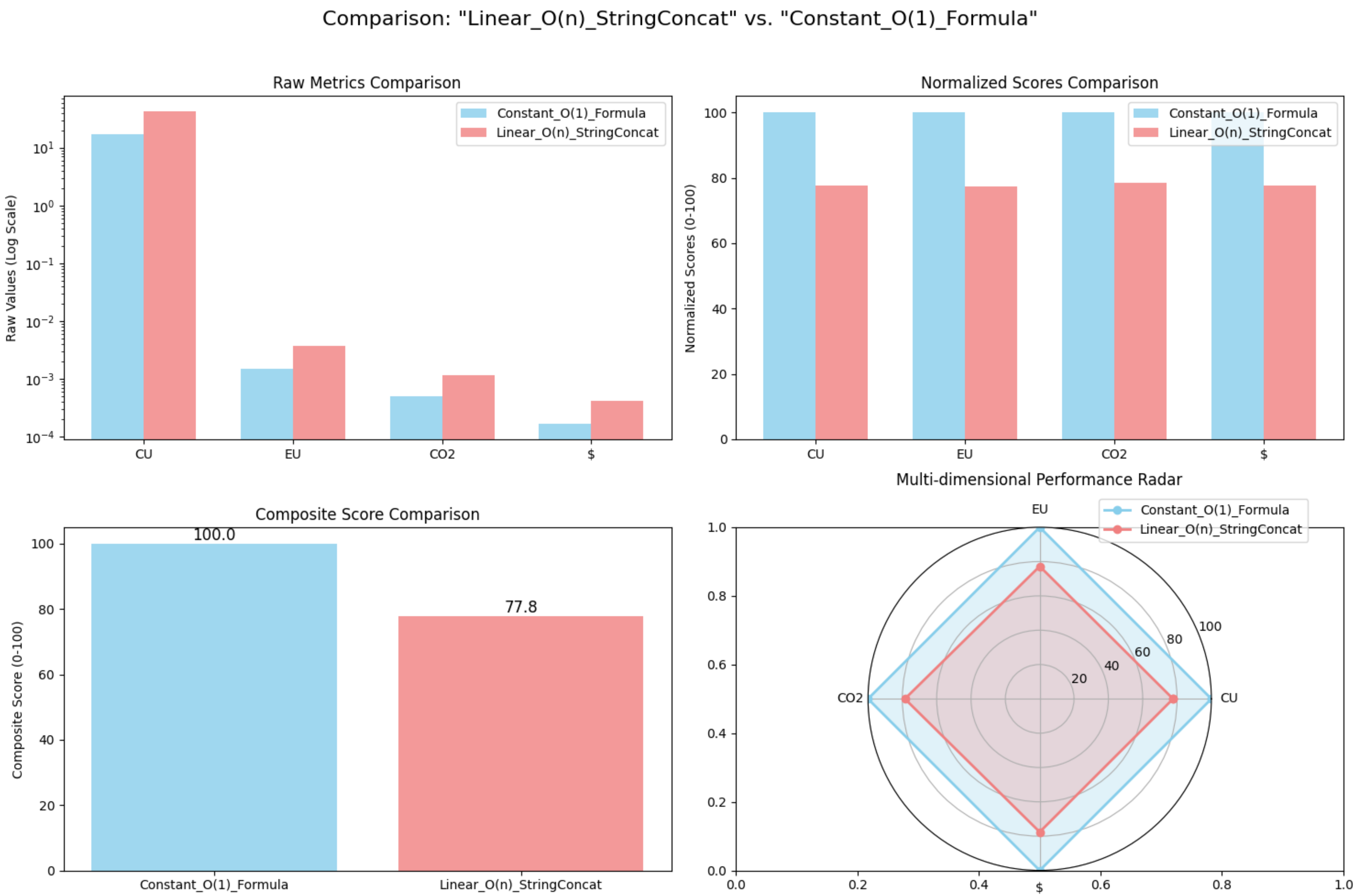


Fig. 6. Generating chart: comparison_Linear_On_StringConcat_vs_Constant_O1_Formula

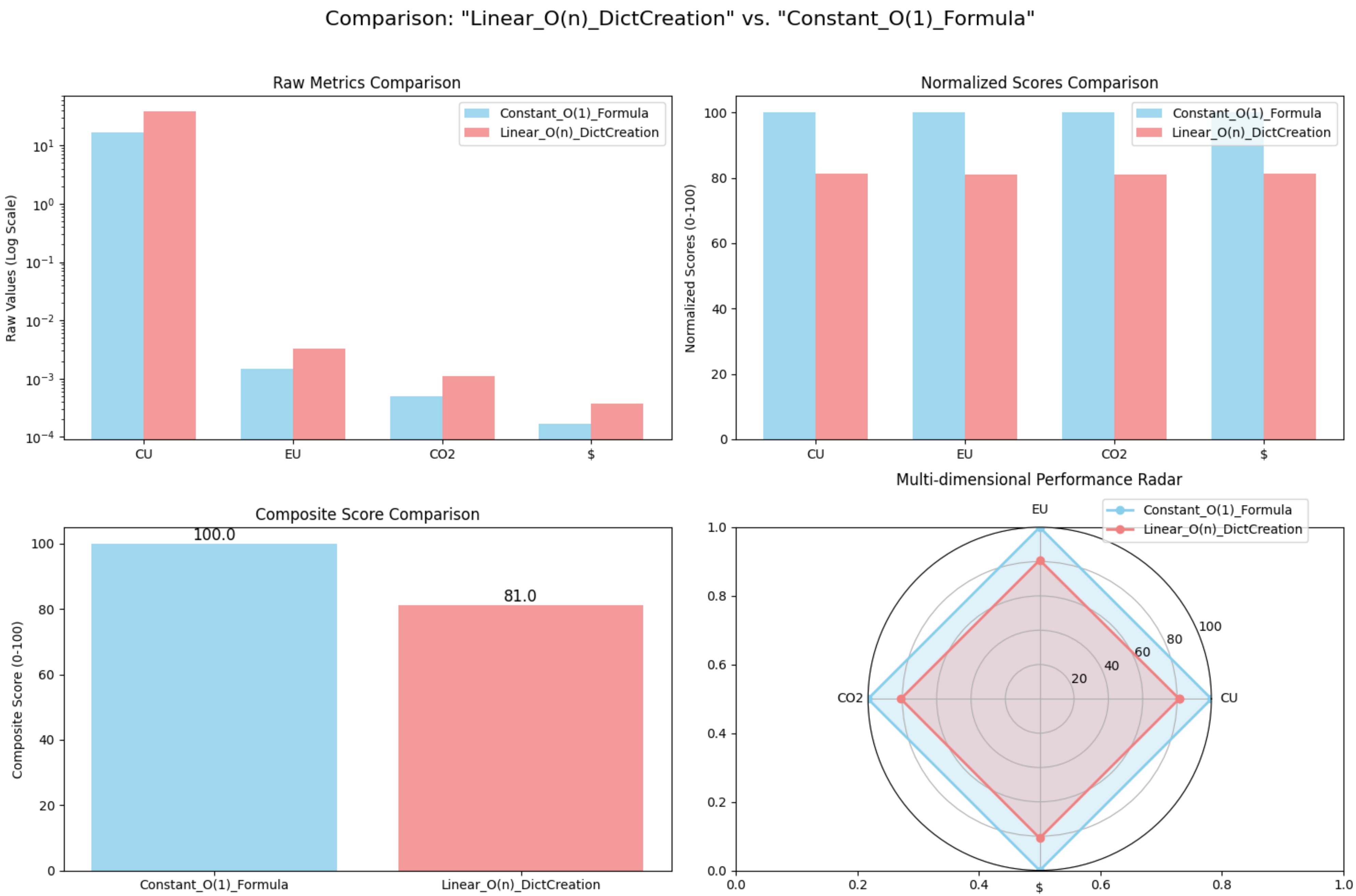


Fig. 7. Generating chart: comparison_Linear_On_DictCreation_vs_Constant_O1_Formula

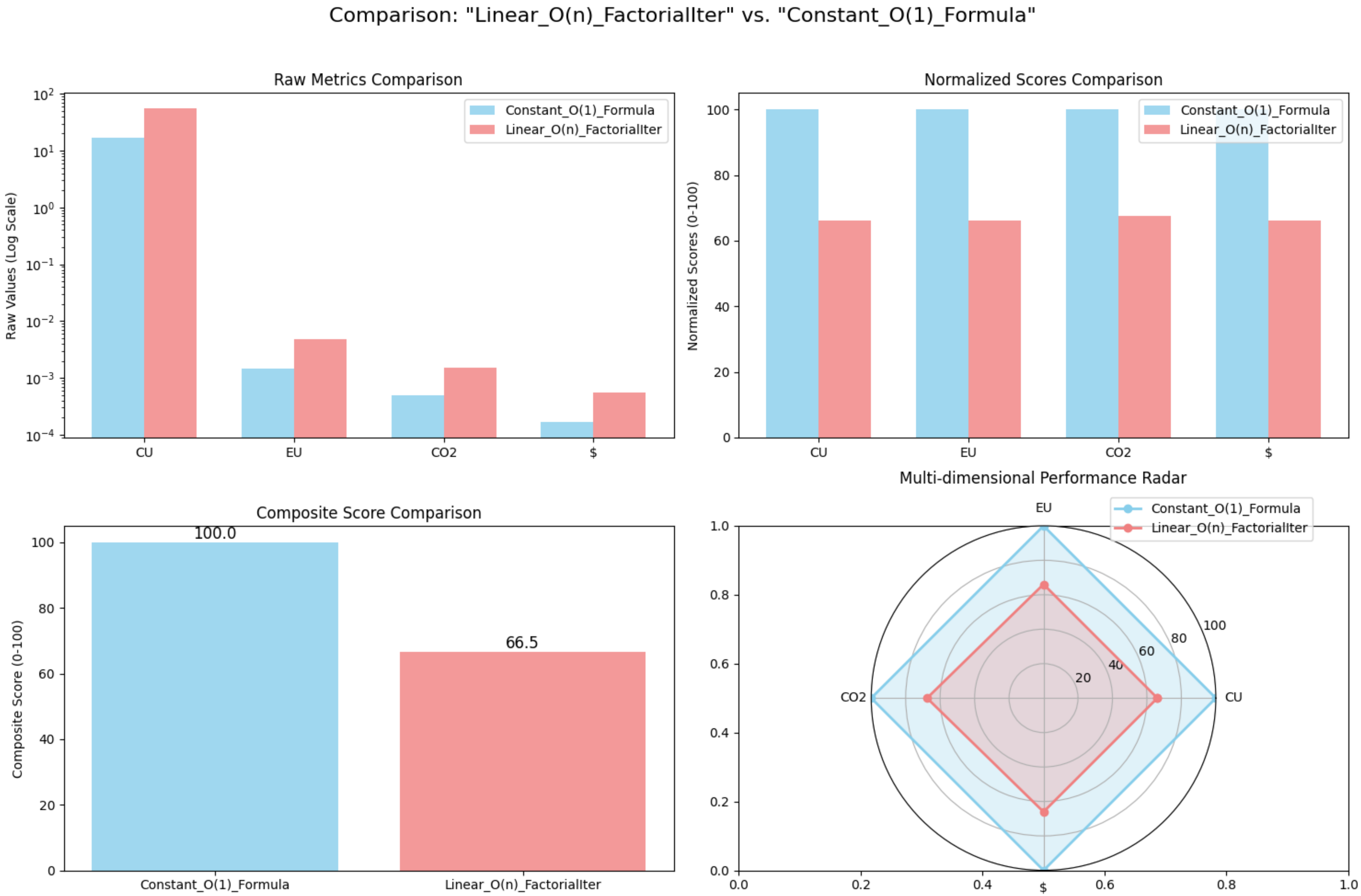


Fig. 8. Generating chart: comparison_Linear_On_FactorialIter_vs_Constant_O1_Formula

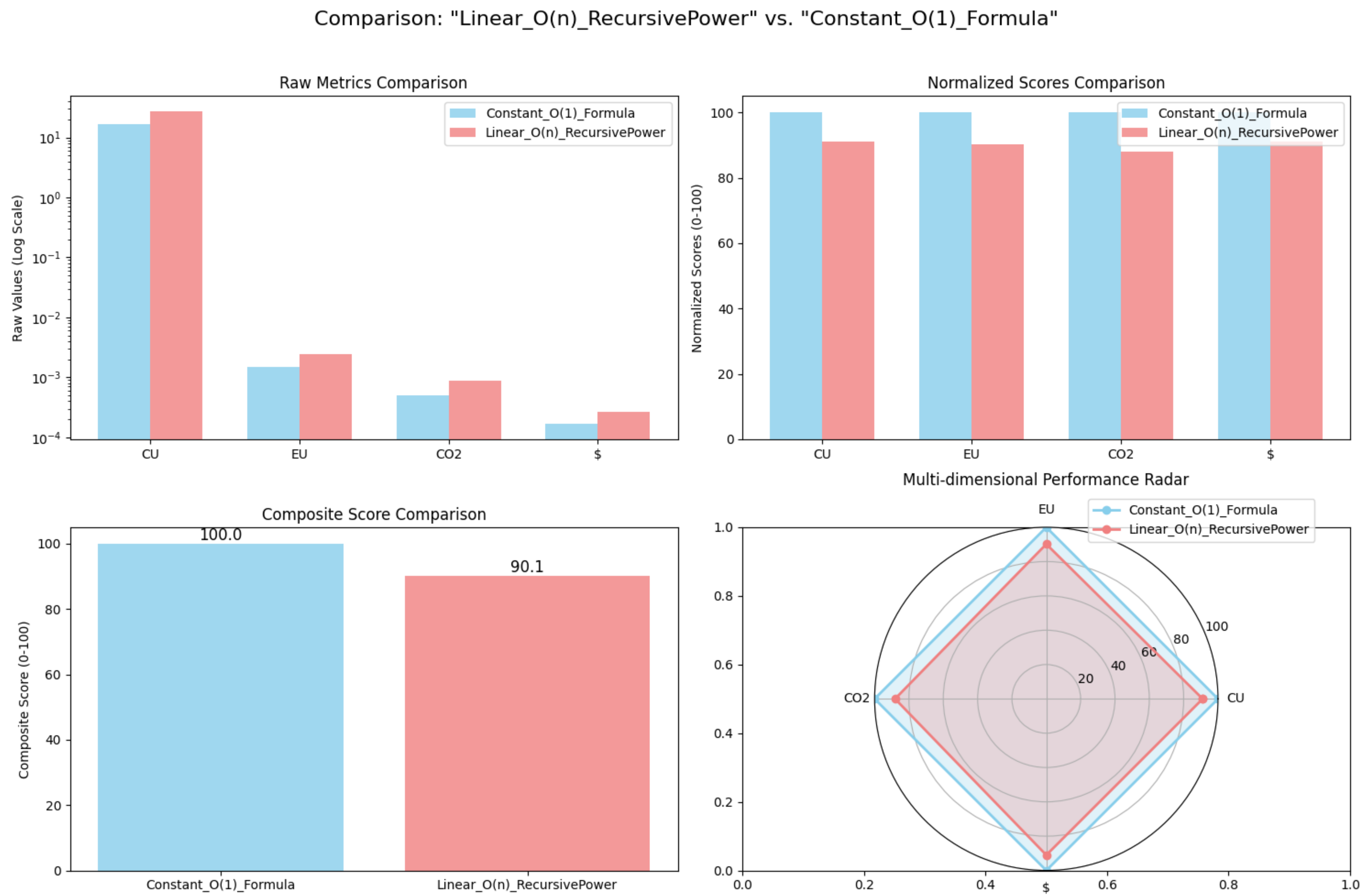


Fig. 9. Generating chart: comparison_Linear_On_RecursivePower_vs_Constant_O1_Formula

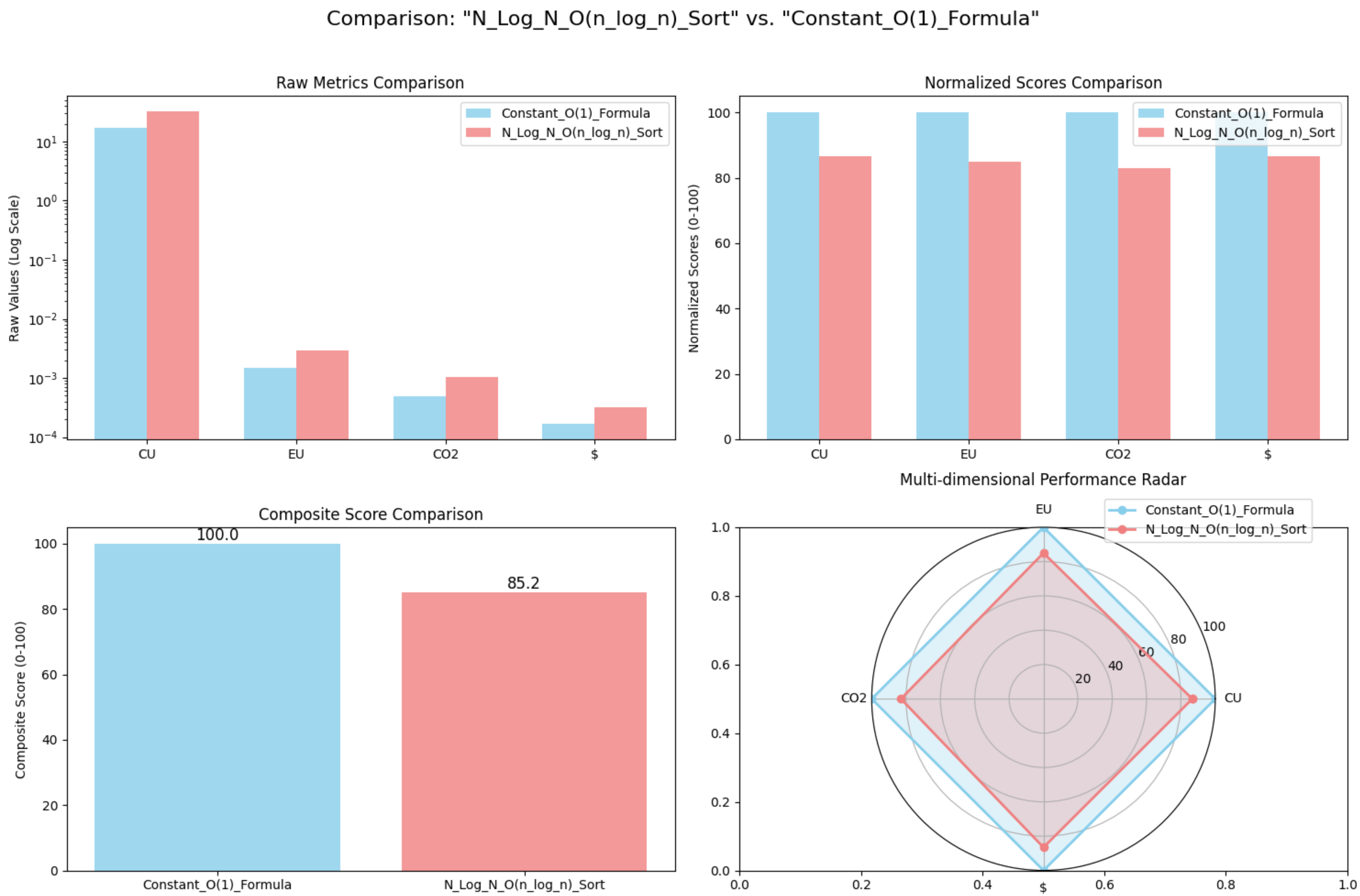


Fig. 10. Generating chart: comparison_N_Log_N_On_log_n_Sort_vs_Constant_O1_Formula

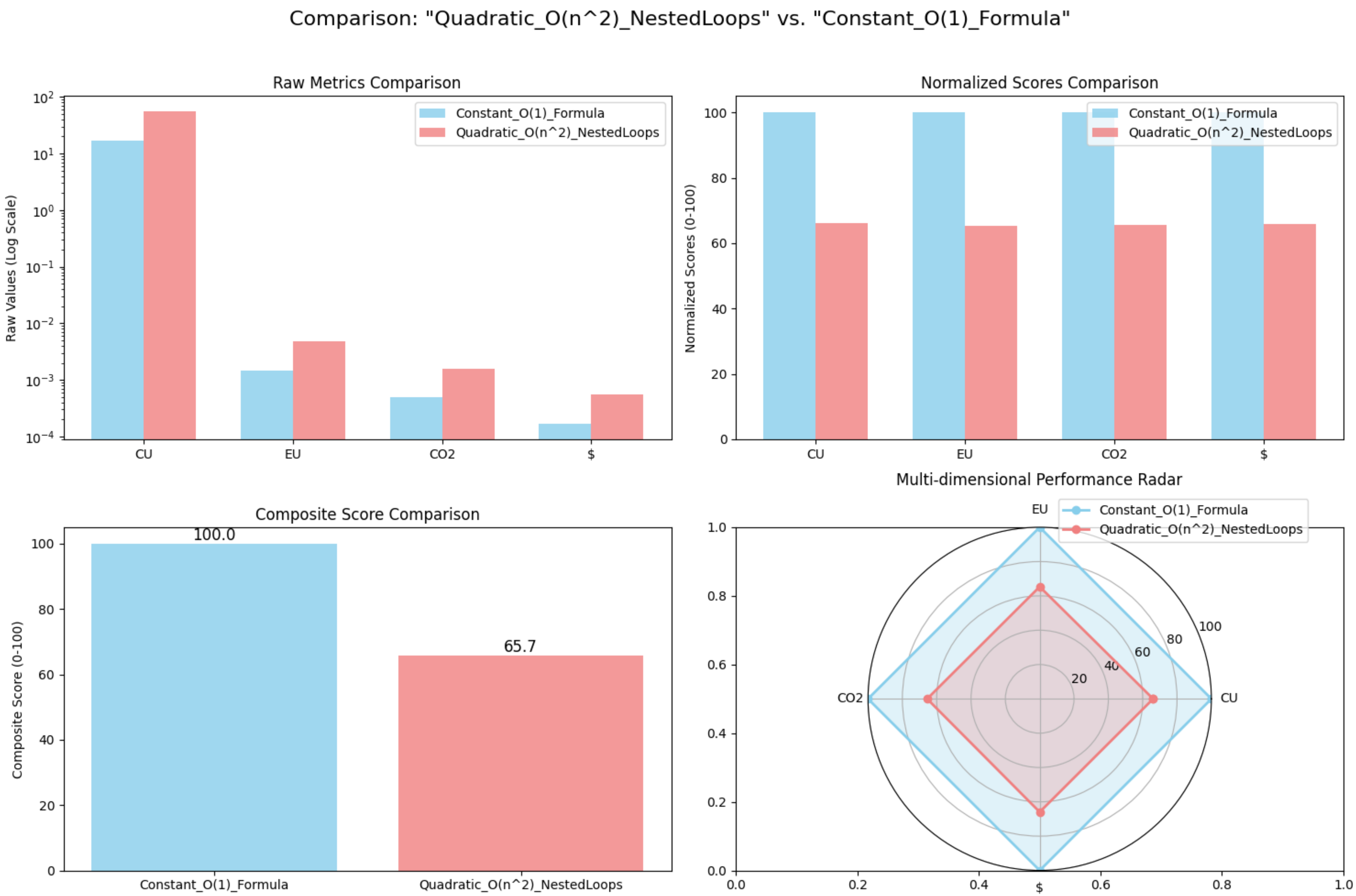


Fig. 11. Generating chart: comparison_Quadratic_On2_NestedLoops_vs_Constant_O1_Formula

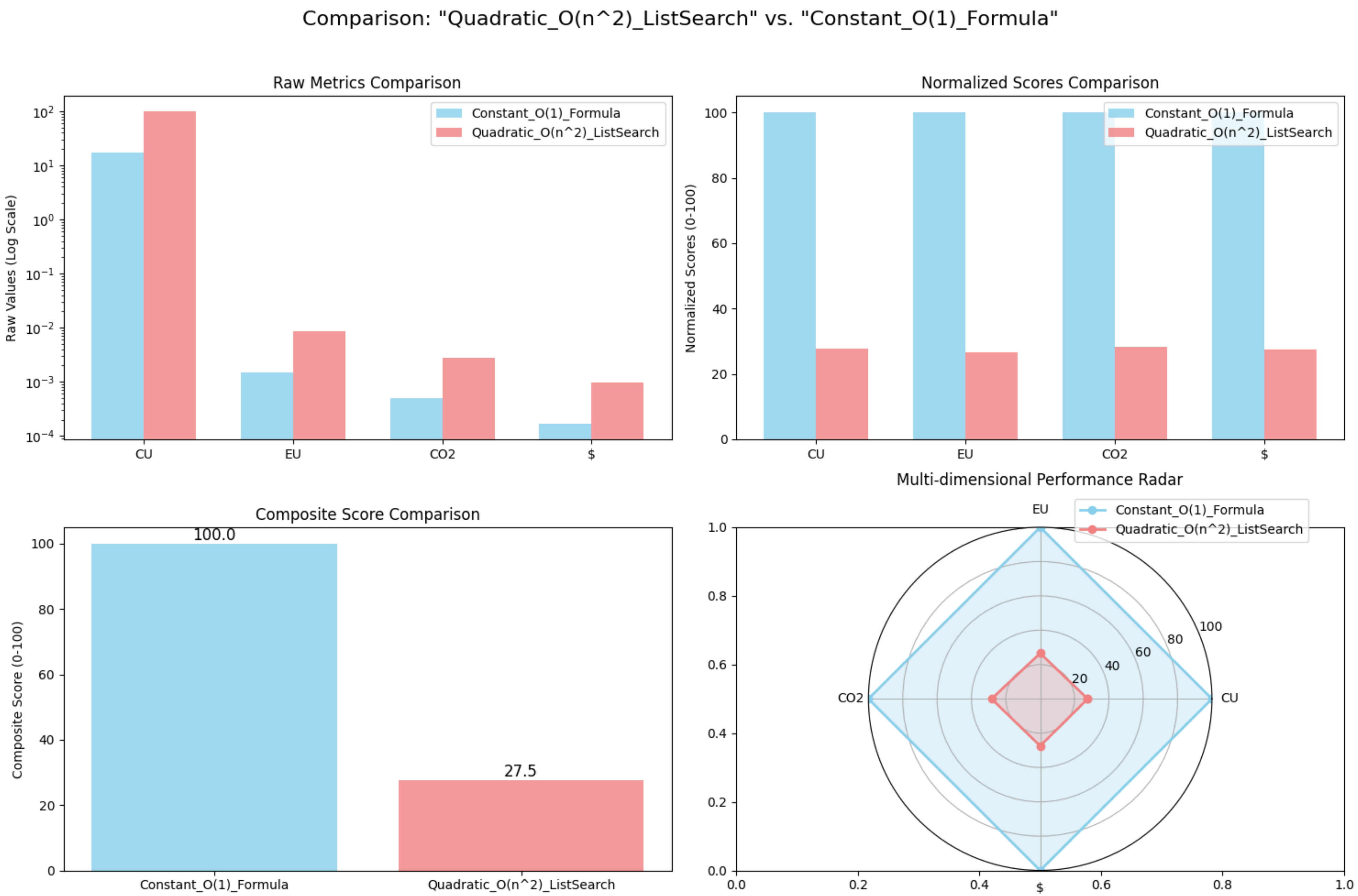


Fig. 12. Generating chart: comparison_Quadratic_On2_ListSearch_vs_Constant_O1_Formula

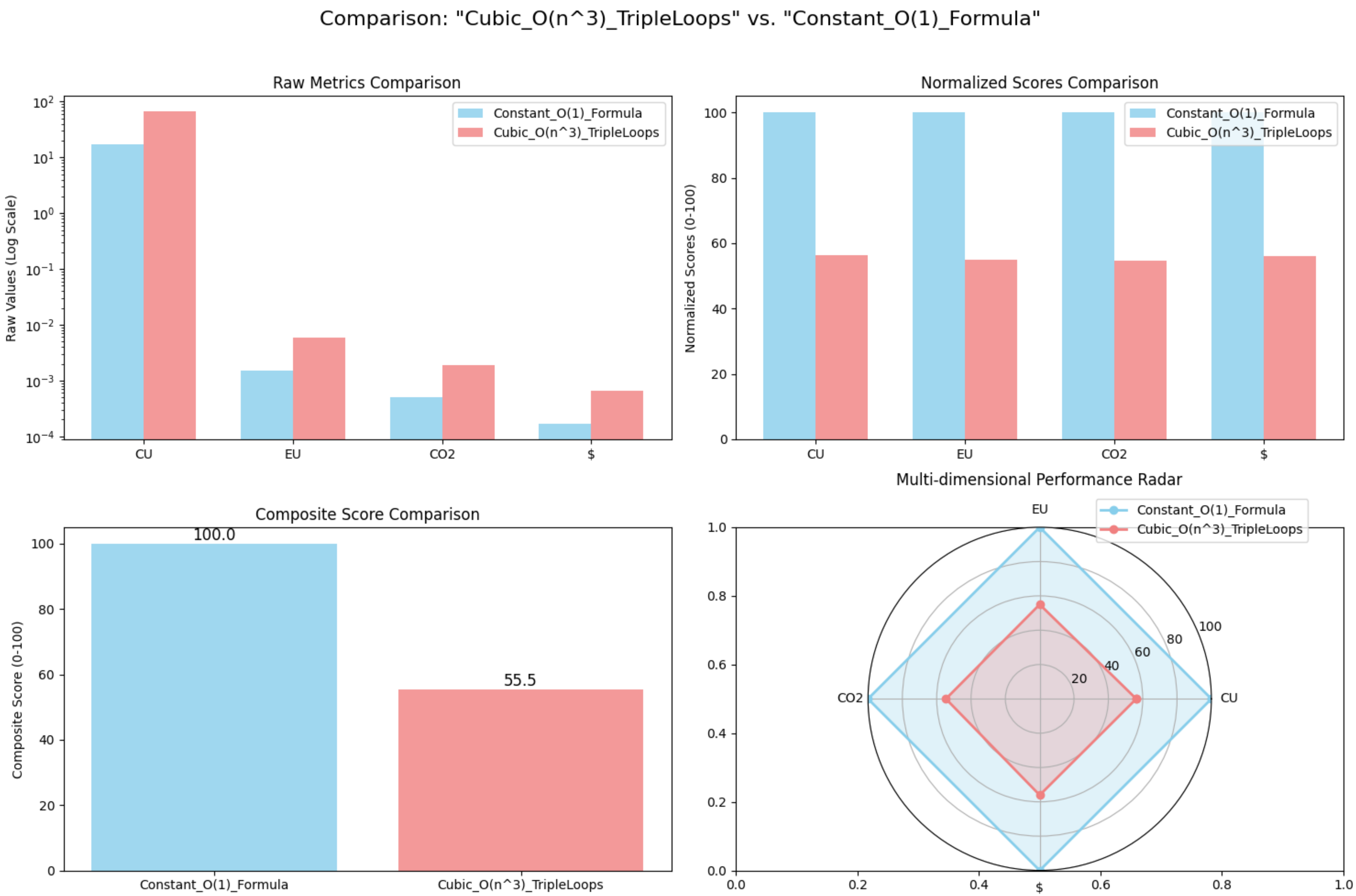


Fig. 13. Generating chart: comparison_Cubic_On3_TripleLoops_vs_Constant_O1_Formula

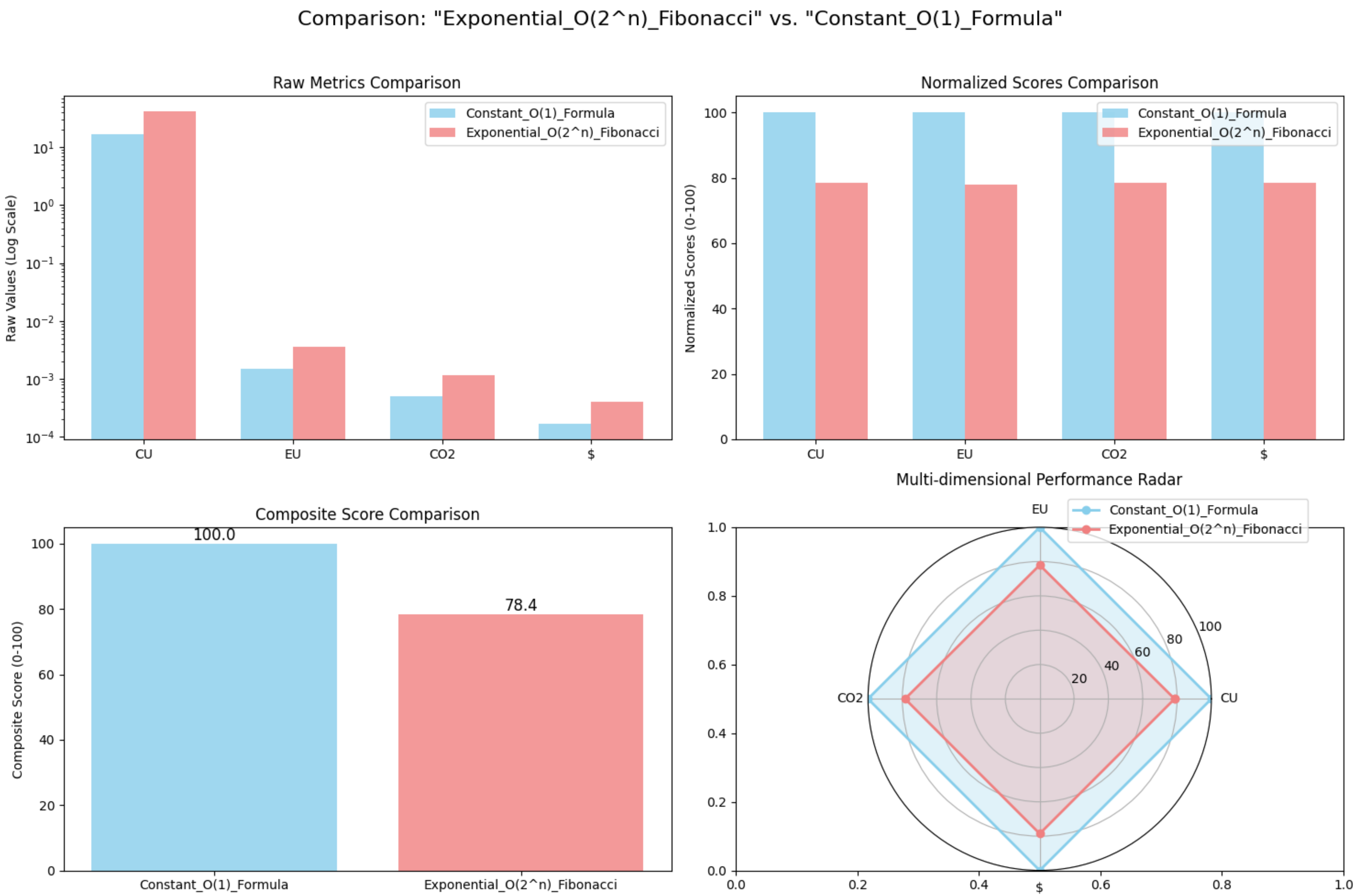


Fig. 14. Generating chart: comparison_Exponential_O2n_Fibonacci_vs_Constant_O1_Formula

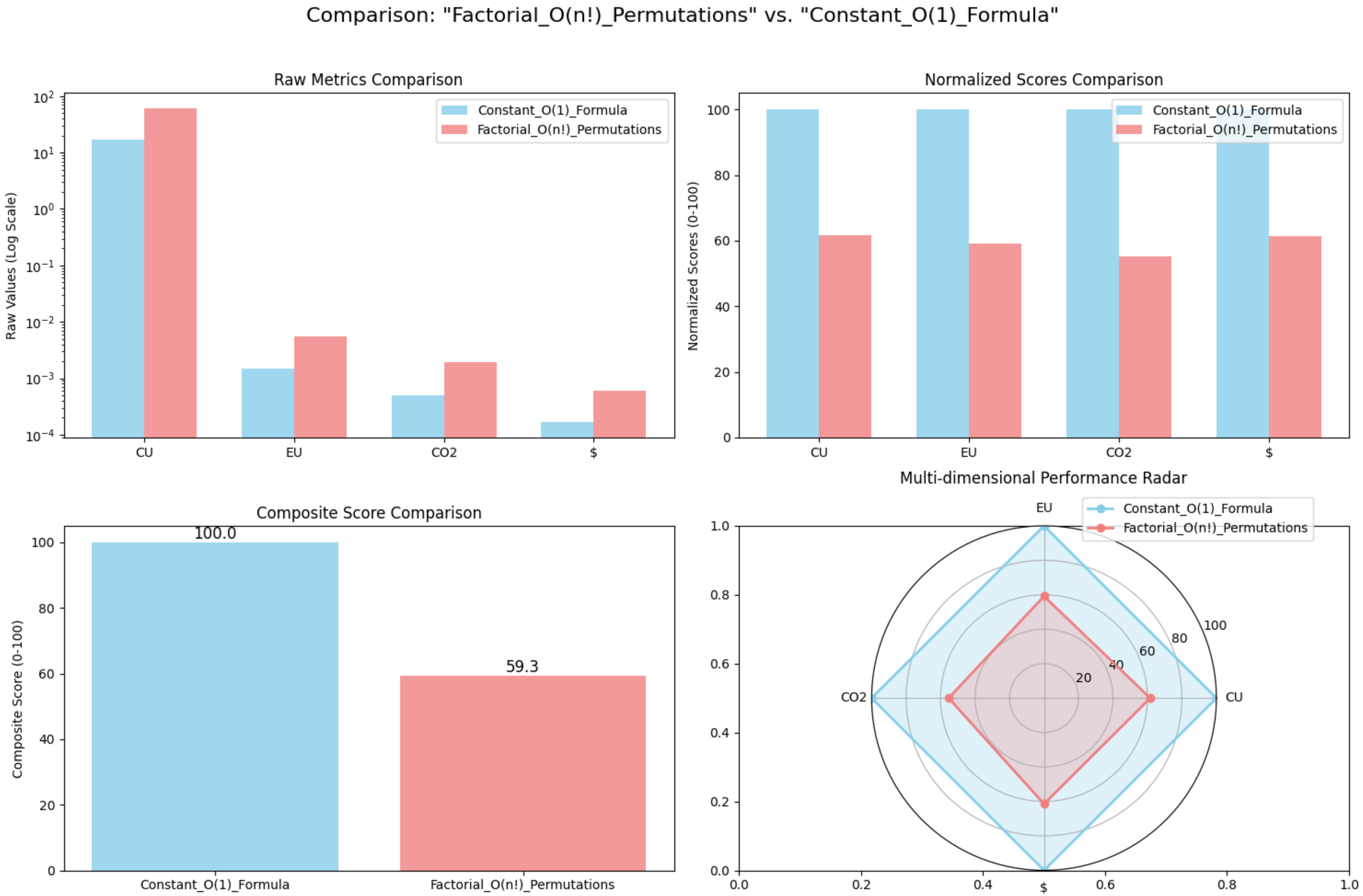


Fig. 15. Generating chart: comparison_Factorial_On_Permutations_vs_Constant_O1_Formula

Table 2. Analyzer configuration and environment

| Configuration Parameter | Value |
|---|---|
| **Detected Architecture** | x86_64 |
| **Available Profiles** | RESEARCH, COMMERCIAL, MOBILE, HPC, DEFAULT |
| **Selected Profile** | RESEARCH |
| **Profile Description** | Research/Academic - focused on performance with environmental awareness |
| **Profile Weights** | {'CU': 0.4, 'EU': 0.3, 'CO2': 0.25, '$': 0.05} |

Table 3. Enhanced algorithm analysis: full suite summary

| Algorithm | CU | EU | $CO_2$ | $ | CU_norm | EU_norm | CO2_norm | $_norm | SCORE GRADE | EFFICIENCY RATING | COMPOSITE SCORE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Constant_O(1)_Formula | 17 | 0.00148 | 0.000498 | 0.000169 | 100 | 100 | 100 | 100 | A+ | Excellent | 100 |
| Linear_O(n)_Sum | 25 | 0.00228 | 0.000763 | 0.000249 | 92.8571 | 91.727 | 91.5578 | 92.8315 | A+ | Excellent | 92.192 |
| Linear_O(n)_RecursivePower | 27 | 0.00243 | 0.000872 | 0.000269 | 91.0714 | 90.1758 | 88.0854 | 91.0394 | A+ | Excellent | 90.0546 |
| N_Log_N_O(n_log_n)_Sort | 32 | 0.00293 | 0.001032 | 0.000319 | 86.6071 | 85.0052 | 82.9882 | 86.5591 | A | Excellent | 85.2194 |

| Algorithm | CU | EU | $CO_2$ | $ | CU_norm | EU_norm | CO2_norm | $_norm | SCORE GRADE | EFFICIENCY RATING | COMPOSITE SCORE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Linear_O(n)_ListAppend | 35 | 0.00313 | 0.001065 | 0.000349 | 83.9286 | 82.9369 | 81.9369 | 83.871 | A- | Good | 83.1303 |
| Linear_O(n)_DictCreation | 38 | 0.00333 | 0.001098 | 0.000379 | 81.25 | 80.8687 | 80.8856 | 81.1828 | A- | Good | 81.0411 |
| Exponential_O(2^n)_ Fibonacci | 41 | 0.00361 | 0.001175 | 0.000408 | 78.5714 | 77.9731 | 78.4326 | 78.5842 | B+ | Good | 78.3579 |
| Linear_O(n)_StringConcat | 42 | 0.00368 | 0.001172 | 0.000419 | 77.6786 | 77.2492 | 78.5282 | 77.5986 | B+ | Good | 77.7582 |
| Linear_O(n)_FactorialIter | 55 | 0.00476 | 0.001515 | 0.000548 | 66.0714 | 66.0807 | 67.6011 | 66.0394 | B- | Average | 66.455 |
| Quadratic_O(n^2)_NestedLoops | 55 | 0.00483 | 0.001579 | 0.000549 | 66.0714 | 65.3568 | 65.5623 | 65.9498 | B- | Average | 65.7237 |
| Factorial_O(n!)_ Permutations | 60 | 0.00543 | 0.001901 | 0.000599 | 61.6071 | 59.152 | 55.3042 | 61.4695 | C | Average | 59.288 |
| Cubic_O(n^3)_ TripleLoops | 66 | 0.00583 | 0.001922 | 0.000659 | 56.25 | 55.0155 | 54.6352 | 56.0932 | C | Average | 55.4681 |
| Quadratic_O(n^2)_ ListSearch | 98 | 0.00858 | 0.002748 | 0.000979 | 27.6786 | 26.577 | 28.3211 | 27.4194 | F | Poor | 27.4958 |
| Logarithmic_O(log_n)_ BinarySearch | 118 | 0.01025 | 0.003209 | 0.001181 | 9.82143 | 9.30714 | 13.6349 | 9.319 | F | Poor | 10.5954 |

| Algorithm | CU | EU | $CO_2$ | $ | CU_norm | EU_norm | CO2_norm | $_norm | SCORE GRADE | EFFICIENCY RATING | COMPOSITE SCORE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sqrt_O(sqrt_n)_PrimalityTest | 129 | 0.01115 | 0.003637 | 0.001285 | 0 | 0 | 0 | 0 | F | Poor | 0 |

| Algorithm | COMPOSITE_SCORE | Score_vs_Best | CU_%_vs_Best | EU_%_vs_Best | CO2_%_vs_Best | $_%_vs_Best |
|---|---|---|---|---|---|---|
| Constant_O(1)_Formula | 100.00 | +0.00 | +0.00% | +0.00% | +0.00% | +0.00% |
| Linear_O(n)_Sum | 92.19 | -7.81 | +47.06% | +54.05% | +53.21% | +47.34% |
| Linear_O(n)_RecursivePower | 90.05 | -9.95 | +58.82% | +64.19% | +75.10% | +59.17% |
| N_Log_N_O(n_log_n)_Sort | 85.22 | -14.78 | +88.24% | +97.97% | +107.23% | +88.76% |
| Linear_O(n)_ListAppend | 83.13 | -16.87 | +105.88% | +111.49% | +113.86% | +106.51% |
| Linear_O(n)_DictCreation | 81.04 | -18.96 | +123.53% | +125.00% | +120.48% | +124.26% |
| Exponential_O(2^n)_Fibonacci | 78.36 | -21.64 | +141.18% | +143.92% | +135.94% | +141.42% |
| Linear_O(n)_StringConcat | 77.76 | -22.24 | +147.06% | +148.65% | +135.34% | +147.93% |
| Linear_O(n)_FactorialIter | 66.46 | -33.54 | +223.53% | +221.62% | +204.22% | +224.26% |
| Quadratic_O(n^2)_NestedLoops | 65.72 | -34.28 | +223.53% | +226.35% | +217.07% | +224.85% |
| Factorial_O(n!)_Permutations | 59.29 | -40.71 | +252.94% | +266.89% | +281.73% | +254.44% |
| Cubic_O(n^3)_TripleLoops | 55.47 | -44.53 | +288.24% | +293.92% | +285.94% | +289.94% |
| Quadratic_O(n^2)_ListSearch | 27.50 | -72.50 | +476.47% | +479.73% | +451.81% | +479.29% |
| Logarithmic_O(log_n)_BinarySearch | 10.60 | -89.40 | +594.12% | +592.57% | +544.38% | +598.82% |
| Sqrt_O(sqrt_n)_PrimalityTest | 0.00 | -100.00 | +658.82% | +653.38% | +630.32% | +660.36% |

Fig. 16. Detailed comparison against the best performing algorithm (Baseline algorithm (highest score): 'Constant_O(1)_Formula')

Let the following lists for **"Function Names"**:
List_1 (complexity_cost_profiler.py) = [CostAnalyzer.__init__, CostAnalyzer.analyze_function, CostAnalyzer.analyze_llvm_ir,

CostAnalyzer.analyze_ptx, CostAnalyzer.compare_functions, CostAnalyzer.fetch_carbon_intensity, InstructionCostModel.__init__, InstructionCostModel._load_weights, InstructionCostModel.get_cost]

List_2 (custom_imputer.py) = [KZImputer.__getstate__, KZImputer.__init__, KZImputer.__repr__, KZImputer.__setstate__, KZImputer.__sklearn_clone__, KZImputer.__sklearn_tags__, KZImputer._check_feature_names, KZImputer._check_n_features, KZImputer._find_nan_blocks, KZImputer._get_doc_link, KZImputer._get_metadata_request, KZImputer._get_tags, KZImputer._impute_1_gap, KZImputer._impute_2_gap, KZImputer._impute_3_gap, KZImputer._impute_4_gap, KZImputer._impute_5_gap, KZImputer._impute_series, KZImputer._more_tags, KZImputer._repr_html_inner, KZImputer._repr_mimebundle_, KZImputer._validate_data, KZImputer._validate_params, KZImputer.evaluate_metrics, KZImputer.fit, KZImputer.fit_transform, KZImputer.generate_synthetic_gaps, KZImputer.get_metadata_routing, KZImputer.get_params, KZImputer.impute_gap, KZImputer.impute_gap_old, KZImputer.set_output, KZImputer.set_params, KZImputer.transform]

List_3 (complexity_cost_profiler_en.py) = [CompositeScoreCalculator.__init__, CompositeScoreCalculator._get_efficiency_rating, CompositeScoreCalculator._get_profile_description, CompositeScoreCalculator._get_score_grade, CompositeScoreCalculator._z_to_percentile, CompositeScoreCalculator.calculate_composite_score, CompositeScoreCalculator.get_profile_info, CompositeScoreCalculator.normalize_metric, CompositeScoreCalculator.update_reference_values, EnhancedCostAnalyzer.__init__, EnhancedCostAnalyzer._calculate_benchmark_stats, EnhancedCostAnalyzer.analyze_function, EnhancedCostAnalyzer.analyze_llvm_ir, EnhancedCostAnalyzer.analyze_ptx, EnhancedCostAnalyzer.benchmark_suite, EnhancedCostAnalyzer.compare_functions, EnhancedCostAnalyzer.fetch_carbon_intensity, InstructionCostModel.__init__, InstructionCostModel._get_bytecode_mapping, InstructionCostModel._get_default_cost_model, InstructionCostModel._load_weights, InstructionCostModel.get_cost, create_benchmark_summary_chart, create_enhanced_comparison_chart, generate_recommendations, main, save_enhanced_csv]

List_4 (ds_tool.py) = [CorrelationConfig.__copy__, CorrelationConfig.__deepcopy__, CorrelationConfig.__delattr__, CorrelationConfig.__eq__, CorrelationConfig.__getattr__, CorrelationConfig.__getstate__, CorrelationConfig.__init__, CorrelationConfig.__iter__, CorrelationConfig.__pretty__, CorrelationConfig.__replace__, CorrelationConfig.__repr__, CorrelationConfig.__repr_args__, CorrelationConfig.__repr_name__, CorrelationConfig.__repr_recursion__, CorrelationConfig.__repr_str__, CorrelationConfig.__rich_repr__, CorrelationConfig.__setattr__, CorrelationConfig.__setstate__, CorrelationConfig.__str__, CorrelationConfig._calculate_keys, CorrelationConfig._copy_and_set_values, CorrelationConfig._iter, CorrelationConfig._setattr_handler, CorrelationConfig.copy, CorrelationConfig.dict, CorrelationConfig.json, CorrelationConfig.model_copy, CorrelationConfig.model_dump, CorrelationConfig.model_dump_json, CorrelationConfig.model_post_init, DSTools.__init__, DSTools.add_missing_value_features, DSTools.calculate_entropy, DSTools.calculate_kl_divergence, DSTools.category_stats, DSTools.chatterjee_correlation, DSTools.check_NINF, DSTools.compute_metrics, DSTools.corr_matrix, DSTools.describe_categorical, DSTools.describe_numeric, DSTools.df_stats, DSTools.evaluate_classification, DSTools.function_list, DSTools.generate_alphanum_codes, DSTools.generate_distribution, DSTools.generate_distribution_from_metrics, DSTools.grubbs_test, DSTools.labeling, DSTools.min_max_scale, DSTools.plot_confusion_matrix, DSTools.read_dataframes_from_zip, DSTools.remove_outliers_iqr, DSTools.save_dataframes_to_zip, DSTools.sparse_calc, DSTools.stat_normal_testing, DSTools.test_stationarity, DSTools.trials_res_df, DSTools.validate_moments, DistributionConfig.__copy__, DistributionConfig.__deepcopy__, DistributionConfig.__delattr__, DistributionConfig.__eq__, DistributionConfig.__getattr__, DistributionConfig.__getstate__, DistributionConfig.__init__, DistributionConfig.__iter__, DistributionConfig.__pretty__, DistributionConfig.__replace__, DistributionConfig.__repr__, DistributionConfig.__repr_args__, DistributionConfig.__repr_name__, DistributionConfig.__repr_recursion__, DistributionConfig.__repr_str__, DistributionConfig.__rich_repr__, DistributionConfig.__setattr__, DistributionConfig.__setstate__, DistributionConfig.__str__, DistributionConfig._calculate_keys, DistributionConfig._copy_and_set_values, DistributionConfig._iter, DistributionConfig._setattr_handler, DistributionConfig.copy, DistributionConfig.dict, DistributionConfig.json, DistributionConfig.model_copy, DistributionConfig.model_dump, DistributionConfig.model_dump_json, DistributionConfig.model_post_init, DistributionConfig.validate_max_greater_than_min, GrubbsTestResult.__copy__, GrubbsTestResult.__deepcopy__, GrubbsTestResult.__delattr__, GrubbsTestResult.__eq__, GrubbsTestResult.__getattr__, GrubbsTestResult.__getstate__, GrubbsTestResult.__init__, GrubbsTestResult.__iter__, GrubbsTestResult.__pretty__, GrubbsTestResult.__replace__, GrubbsTestResult.__repr__, GrubbsTestResult.__repr_args__, GrubbsTestResult.__repr_name__, GrubbsTestResult.__repr_recursion__, GrubbsTestResult.__repr_str__, GrubbsTestResult.__rich_repr__, GrubbsTestResult.__setattr__, GrubbsTestResult.__setstate__, GrubbsTestResult.__str__, GrubbsTestResult._calculate_keys, GrubbsTestResult._copy_and_set_values, GrubbsTestResult._iter, GrubbsTestResult._setattr_handler, GrubbsTestResult.copy, GrubbsTestResult.dict, GrubbsTestResult.json, GrubbsTestResult.model_copy, GrubbsTestResult.model_dump, GrubbsTestResult.model_dump_json, GrubbsTestResult.model_post_init, MetricsConfig.__copy__, MetricsConfig.__deepcopy__, MetricsConfig.__delattr__, MetricsConfig.__eq__, MetricsConfig.__getattr__, MetricsConfig.__getstate__, MetricsConfig.__init__, MetricsConfig.__iter__, MetricsConfig.__pretty__, MetricsConfig.__replace__, MetricsConfig.__repr__, MetricsConfig.__repr_args__, MetricsConfig.__repr_name__, MetricsConfig.__repr_recursion__, MetricsConfig.__repr_str__, MetricsConfig.__rich_repr__, MetricsConfig.__setattr__, MetricsConfig.__setstate__, MetricsConfig.__str__, MetricsConfig._calculate_keys, MetricsConfig._copy_and_set_values, MetricsConfig._iter, MetricsConfig._setattr_handler, MetricsConfig.copy, MetricsConfig.dict, MetricsConfig.json, MetricsConfig.model_copy, MetricsConfig.model_dump, MetricsConfig.model_dump_json, MetricsConfig.model_post_init, OutlierConfig.__copy__, OutlierConfig.__deepcopy__, OutlierConfig.__delattr__, OutlierConfig.__eq__, OutlierConfig.__getattr__, OutlierConfig.__getstate__, OutlierConfig.__init__, OutlierConfig.__iter__, OutlierConfig.__pretty__, OutlierConfig.__replace__, OutlierConfig.__repr__, OutlierConfig.__repr_args__, OutlierConfig.__repr_name__, OutlierConfig.__repr_recursion__, OutlierConfig.__repr_str__, OutlierConfig.__rich_repr__, OutlierConfig.__setattr__, OutlierConfig.__setstate__, OutlierConfig.__str__, OutlierConfig._calculate_keys, OutlierConfig._copy_and_set_values, OutlierConfig._iter, OutlierConfig._setattr_handler, OutlierConfig.copy, OutlierConfig.dict, OutlierConfig.json, OutlierConfig.model_copy, OutlierConfig.model_dump, OutlierConfig.model_dump_json, OutlierConfig.model_post_init]

List_5 (s3_act_function.py) = [_compute_activation, _compute_derivative, smooth_s3_activation]

Table 4. Analysis of external source files (LLVM, PTX, PY)

| File Name | File Type | Function Name | COMPOSITE SCORE | SCORE GRADE | CU | EU | CO2 | $ |
|---|---|---|---|---|---|---|---|---|
| test.ptx | PTX GPU | - | 99.4093 | A+ | 15 | 0.0015 | 0.00075 | 0.00015 |
| file_io.ll | LLVM IR | - | 98.2849 | A+ | 42 | 0.004 | 0.001641 | 0.00042 |
| fibonacci.ll | LLVM IR | - | 96.172 | A+ | 93 | 0.0083 | 0.002939 | 0.00093 |
| str_cat.ll | LLVM IR | - | 92.1819 | A+ | 189 | 0.01705 | 0.005821 | 0.00189 |
| s4_act_function.py | Python | s4 | 89.8761 | A | 245 | 0.02121 | 0.006749 | 0.002448 |

| File Name | File Type | Function Name | COMPOSITE SCORE | SCORE GRADE | CU | EU | CO2 | $ |
|---|---|---|---|---|---|---|---|---|
| generate_instr_models.py | Python | enrich, generate_models | 87.9534 | A | 291 | 0.02563 | 0.008528 | 0.00291 |
| s3_act_function.py | Python | List_5 | 87.6258 | A | 299 | 0.02627 | 0.008592 | 0.002986 |
| matrixMul_kernel_32.ptx | PTX GPU | - | 74.2815 | B | 612 | 0.0612 | 0.0306 | 0.00612 |
| complexity_cost_profiler.py | Python | List_1 | 62.6856 | C+ | 898 | 0.08056 | 0.028342 | 0.008977 |
| custom_imputer.py | Python | List_2 | 50.0176 | C- | 6611 | 0.58633 | 0.196669 | 0.066143 |
| complexity_cost_profiler_en.py | Python | List_3 | 49.524 | D | 6941 | 0.61612 | 0.206337 | 0.069394 |
| ds_tool.py | Python | List_4 | 29.7495 | F | 24725 | 2.20735 | 0.745148 | 0.247147 |

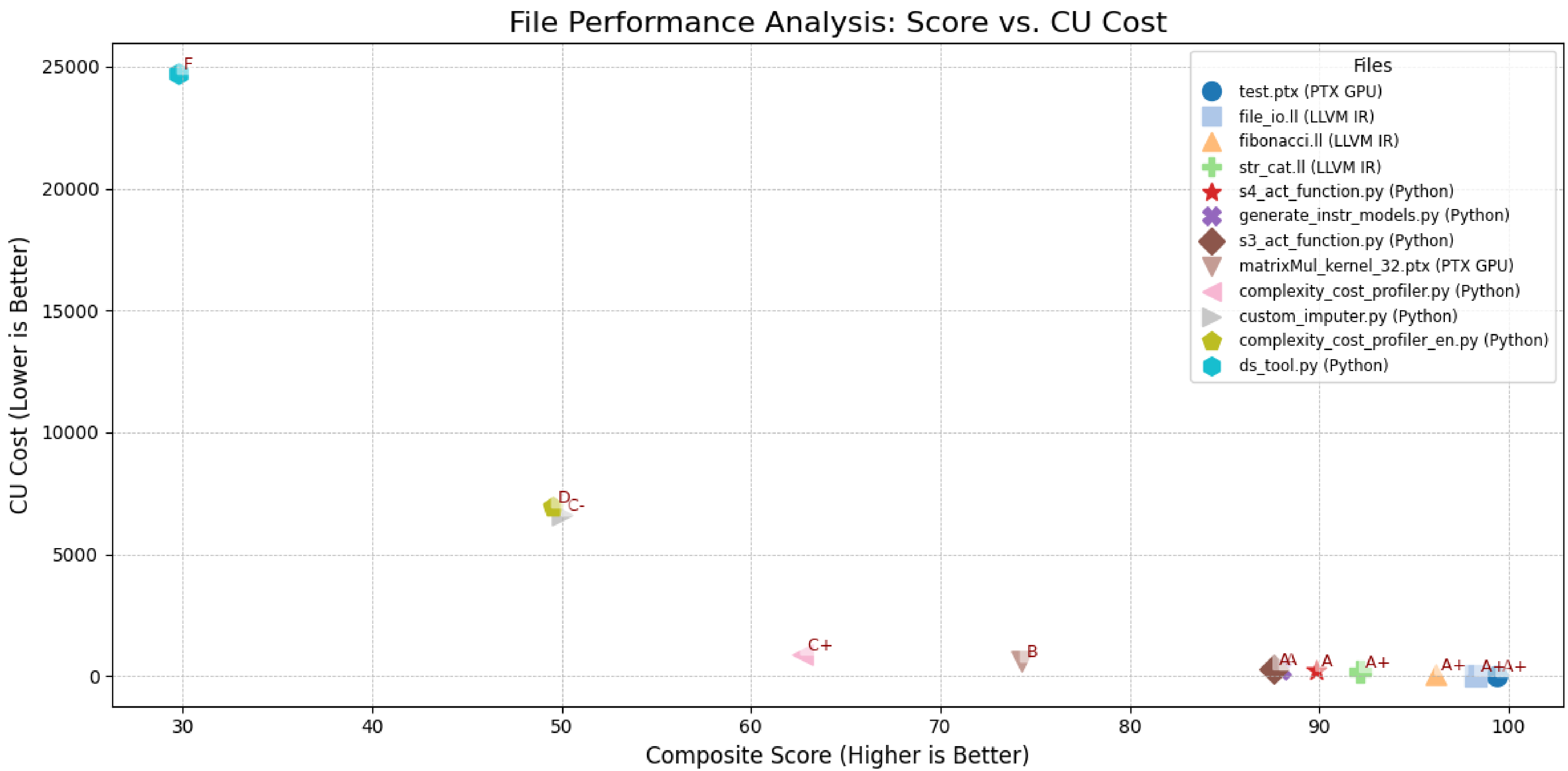


Fig. 17. Displaying top 12 files sorted by COMPOSITE_SCORE (for **CU**-metric, based on Table 4)

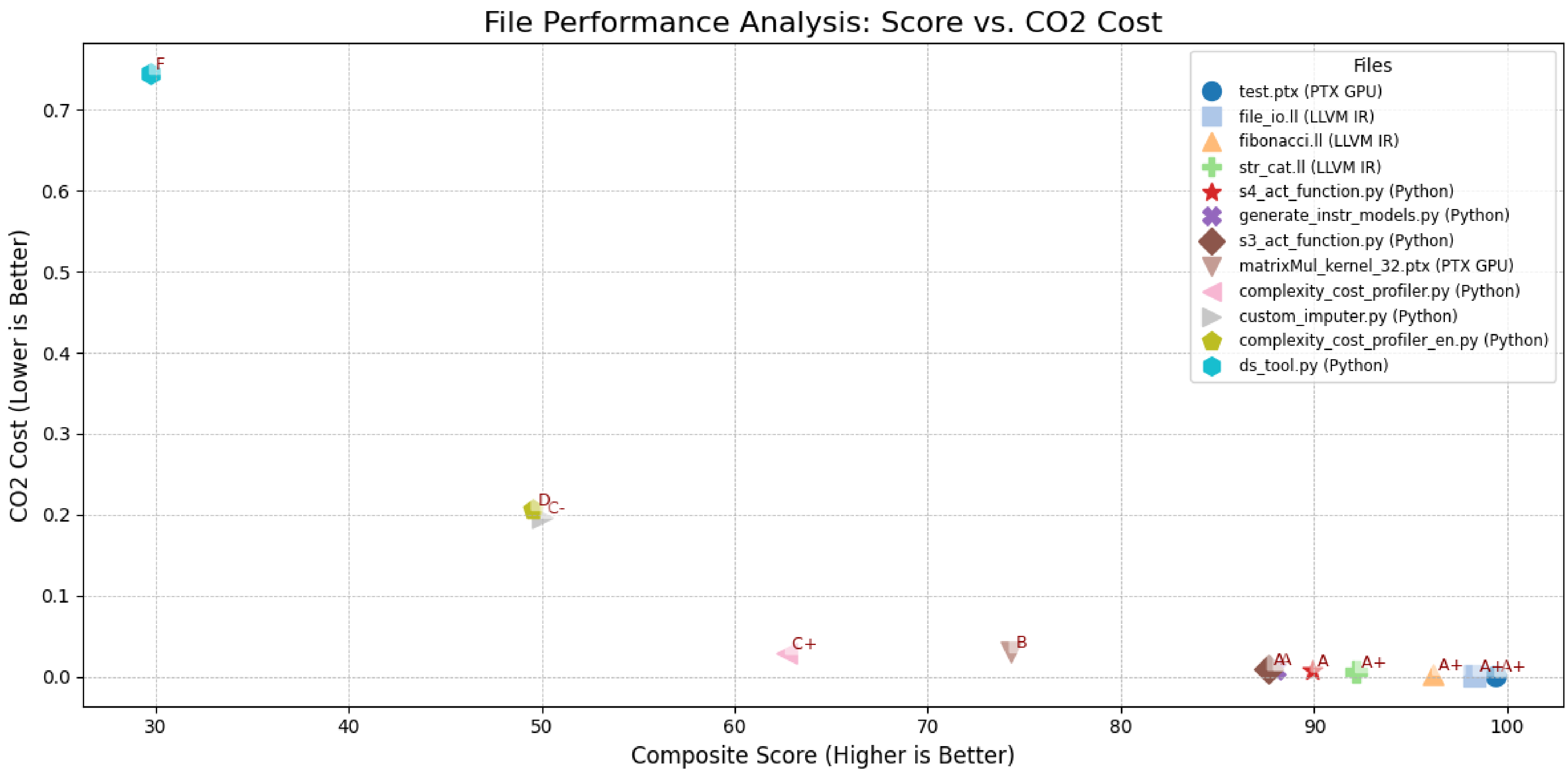


Fig. 18. Displaying top 12 files sorted by COMPOSITE_SCORE (for **$CO_2$**-metric, based on Table 4)


Fig. 19. Displaying top 12 files sorted by COMPOSITE_SCORE (for $-metric, based on Table 4)

Table 5. Analysis of external repository (LLVM, PTX, PY): aggregated assessment for repository: 'kz_data_imputation' [1]

| PROFILE NAME | File Type | Function Name | COMPOSITE_SCORE | SCORE_GRADE | CU | EU | CO2 | $ |
|---|---|---|---|---|---|---|---|---|
| RESEARCH | Python (12) | 89 | 42.23 | D | 12,607 | 1.1194 | 0.3763 | 0.1261 |
| COMMERCIAL | Python (12) | 89 | 30.24 | F | 12,607 | 1.1194 | 0.3763 | 0.1261 |
| MOBILE | Python (12) | 89 | 53.76 | C- | 12,607 | 1.1194 | 0.3763 | 0.1261 |
| HPC | Python (12) | 89 | 36.00 | F | 12,607 | 1.1194 | 0.3763 | 0.1261 |
| DEFAULT | Python (12) | 89 | 37.79 | F | 12,607 | 1.1194 | 0.3763 | 0.1261 |
| TOTAL | All Files (12) | 89 | 40.00 | D | 12,607 | 1.1194 | 0.3763 | 0.1261 |

Table 6. Analysis of external repository (LLVM, PTX, PY): aggregated assessment for repository: 'ds_tools' [1]

| PROFILE NAME | File Type | Function Name | COMPOSITE_SCORE | SCORE_GRADE | CU | EU | CO2 | $ |
|---|---|---|---|---|---|---|---|---|
| RESEARCH | Python (30) | 272 | 22.58 | F | 31,540 | 2.8188 | 0.9585 | 0.3152 |
| COMMERCIAL | Python (30) | 272 | 15.19 | F | 31,540 | 2.8188 | 0.9585 | 0.3152 |
| MOBILE | Python (30) | 272 | 36.53 | F | 31,540 | 2.8188 | 0.9585 | 0.3152 |
| HPC | Python (30) | 272 | 22.17 | F | 31,540 | 2.8188 | 0.9585 | 0.3152 |
| DEFAULT | Python (30) | 272 | 18.99 | F | 31,540 | 2.8188 | 0.9585 | 0.3152 |
| TOTAL | All Files (30) | 272 | 23.09 | F | 31,540 | 2.8188 | 0.9585 | 0.3152 |

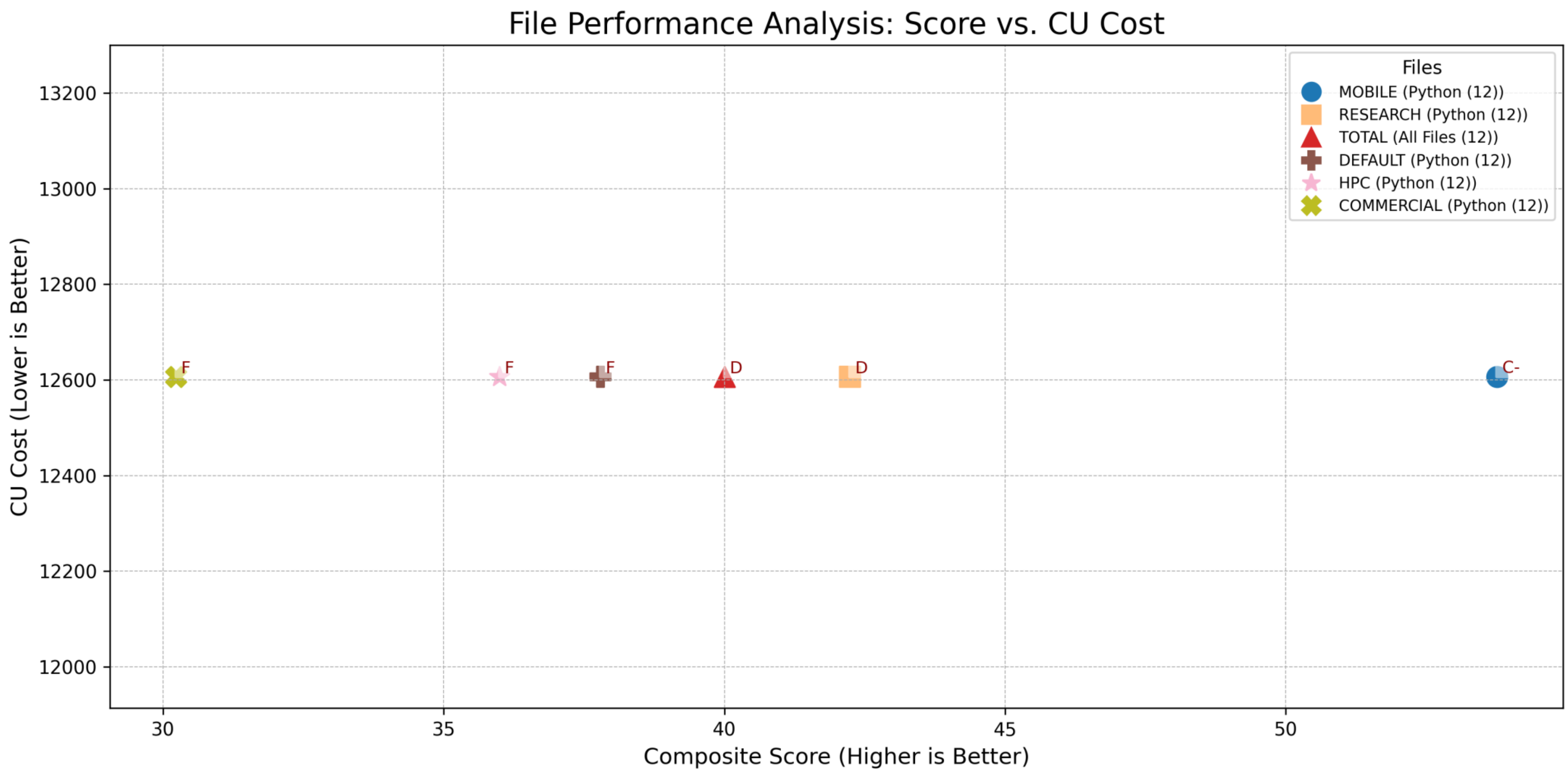


Fig. 20. Displaying repo performance analysis result (for **CU**-metric, based on Table 5) for all profiles

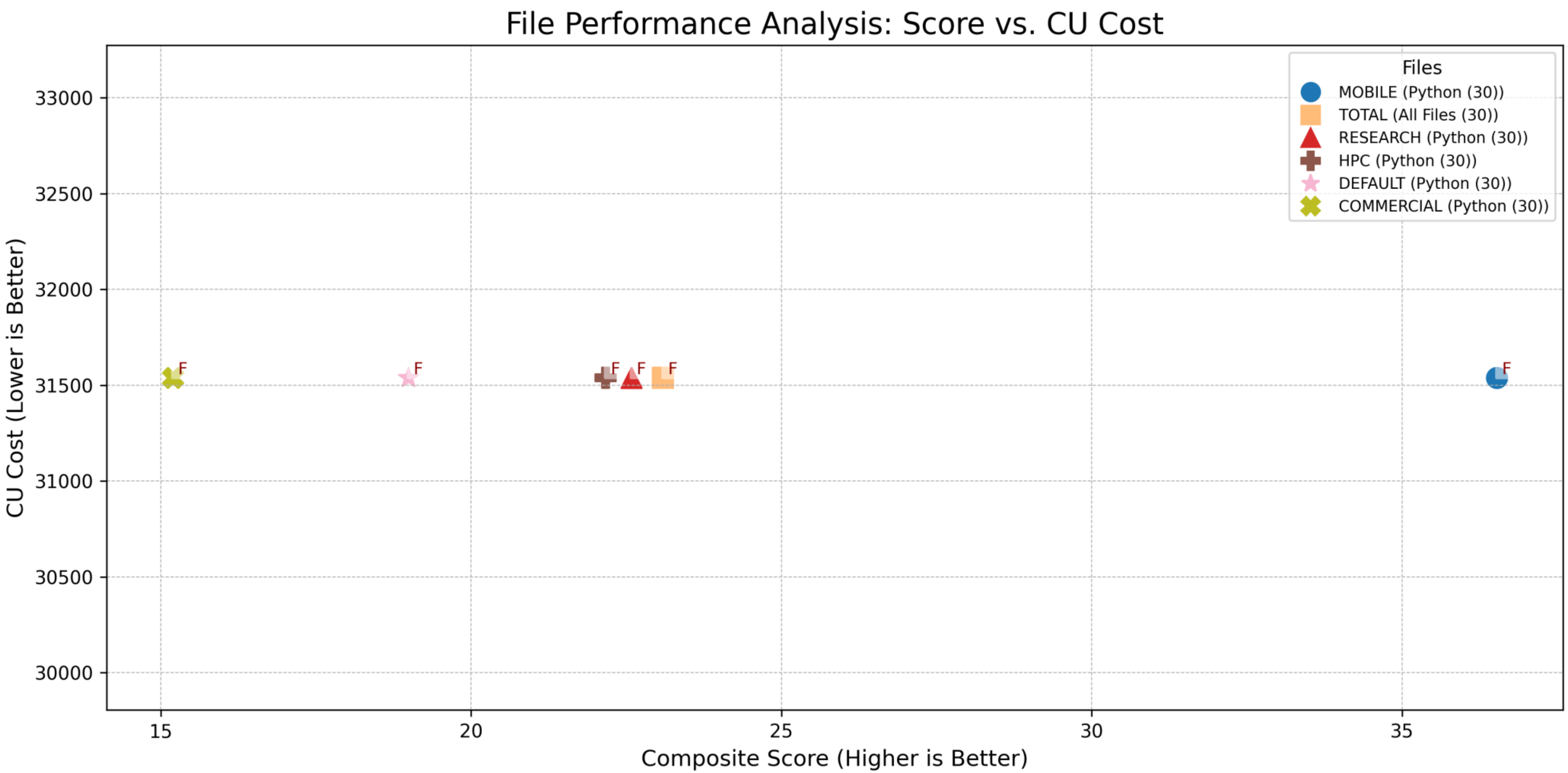


Fig. 21. Displaying repo performance analysis result (for **CU**-metric, based on Table 6) for all profiles

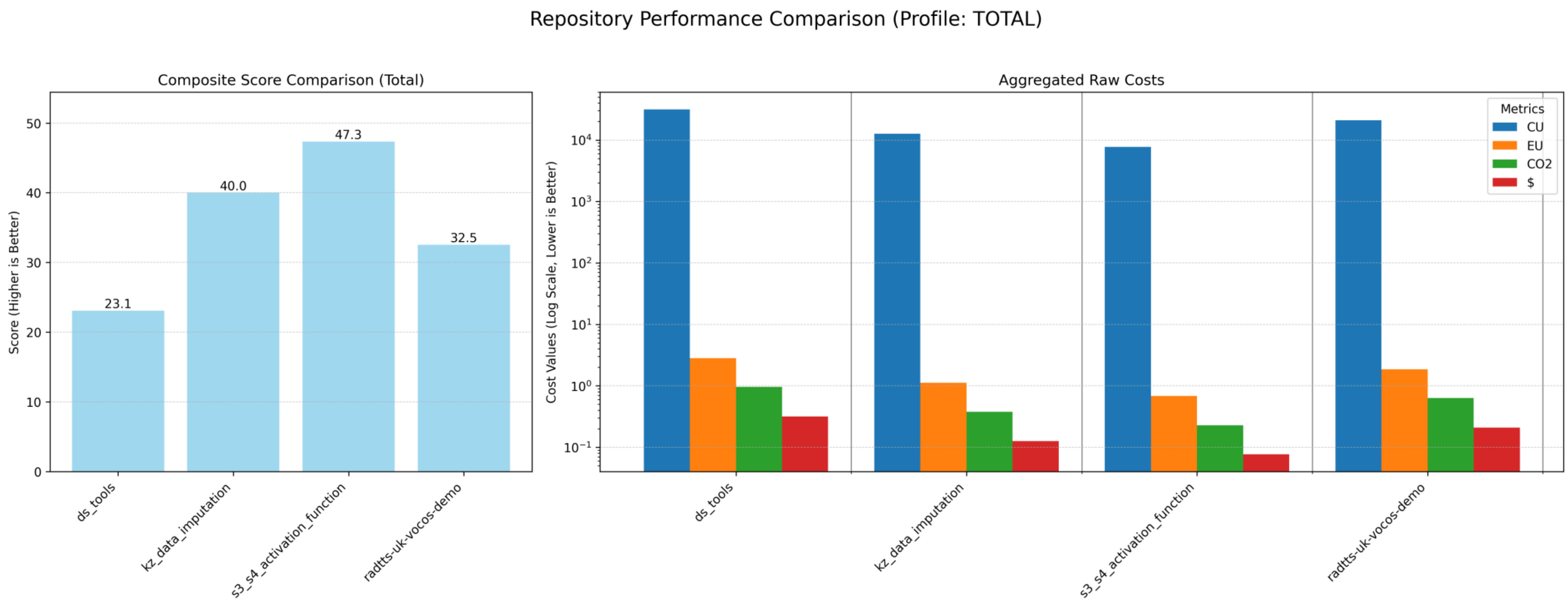


Fig. 22. Displaying repo performance analysis result (for 4 repositories) for all profiles

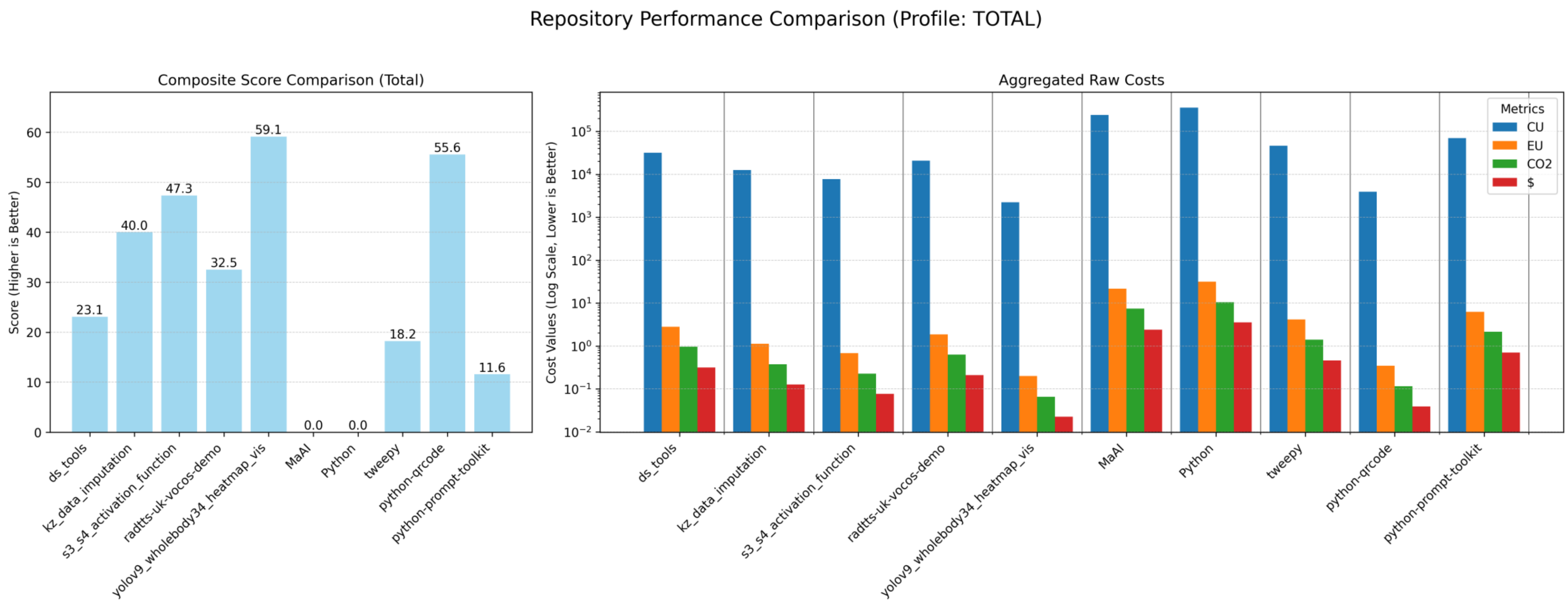


Fig. 23. Displaying repo performance analysis result (for 10 repositories) for all profiles

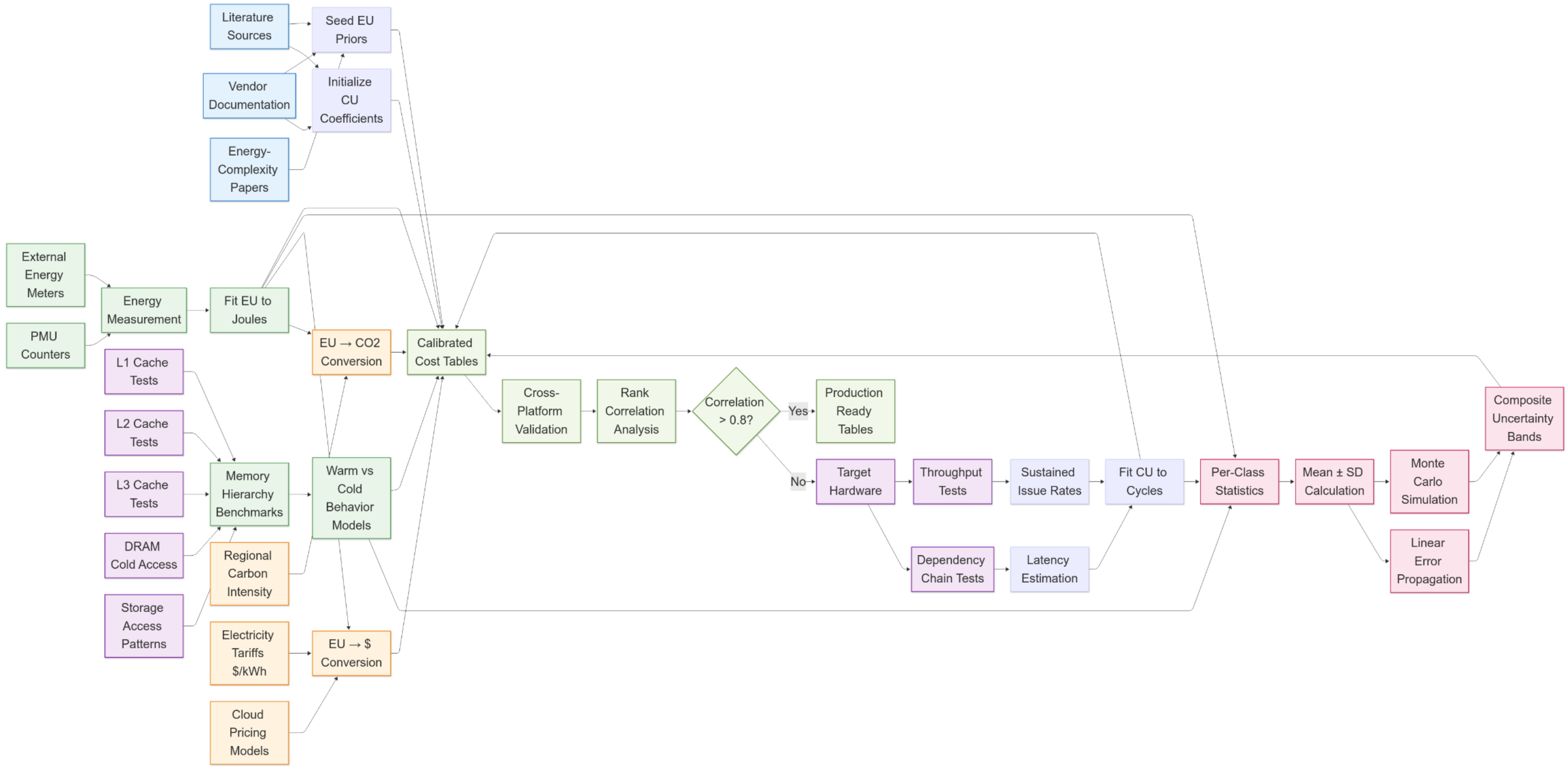


Fig. 24. Multi-architecture calibration protocol architecture

The diagram shows:

1. Five Main Phases: literature priors (blue) - initial coefficient seeding from academic sources; microbenchmarking (purple) - hardware-specific latency and throughput testing; memory hierarchy (green) - cache behavior characterization across levels; dimension mapping (orange) - converting energy to $CO_2$ and monetary costs; uncertainty quantification (pink) - statistical analysis and error propagation.

2. Key Features: validation loop with correlation thresholds for quality assurance; multiple measurement sources (PMU, external meters, various cache levels); regional/temporal factors (carbon intensity, electricity pricing); statistical rigor (mean±SD, error propagation, Monte Carlo); production readiness gate based on validation results.
3. Color-coded components make it easy to follow different aspects: literature sources and documentation; hardware testing and measurement; energy/performance data collection; cost dimension conversions; statistical analysis steps; final output and validation.

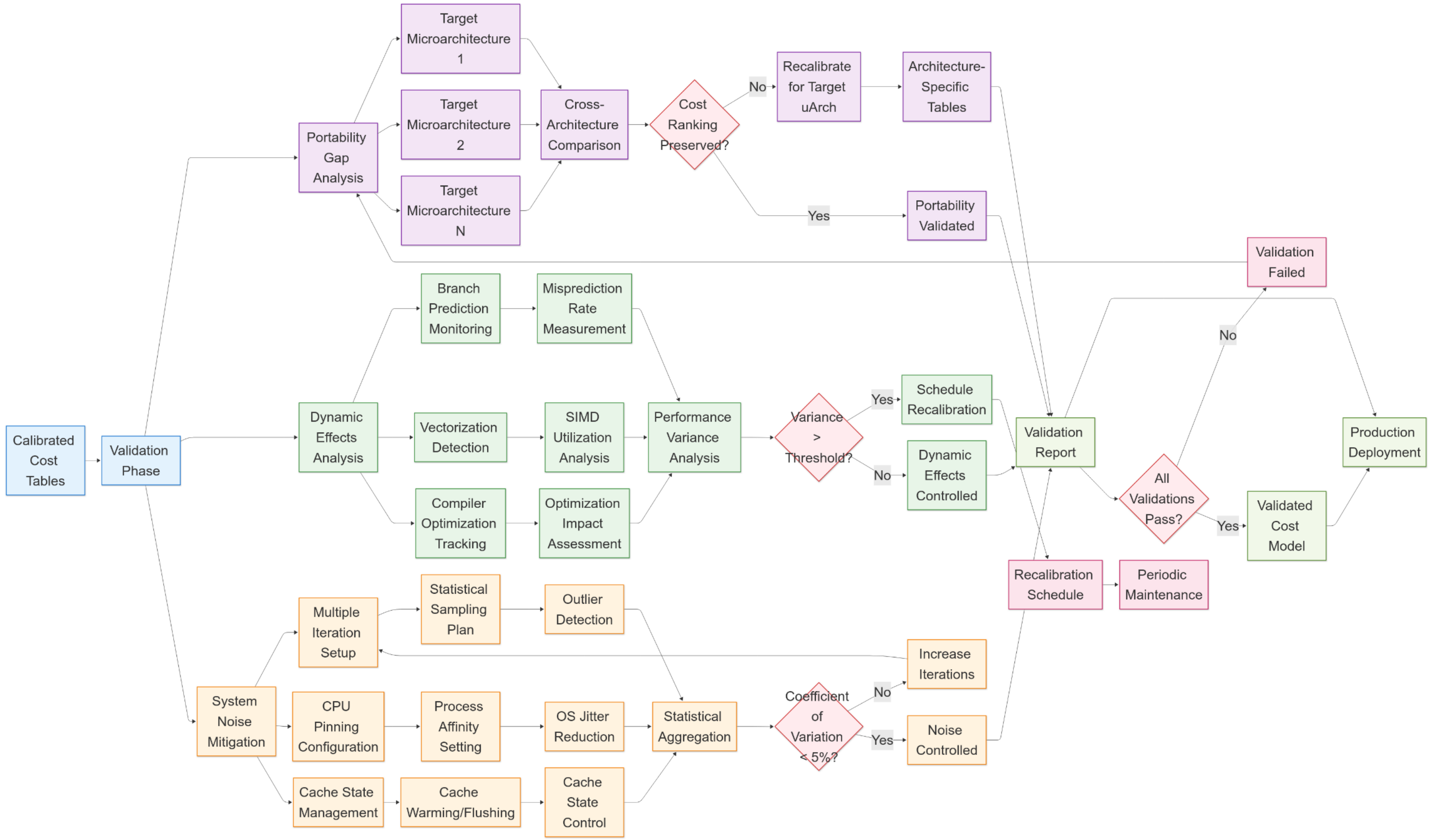


Fig. 25. Comprehensive validation phase diagram

The diagram shows:

1. Three parallel validation streams: portability gap analysis (purple); cross-architecture comparison testing; ranking preservation validation; architecture-specific recalibration when needed.
2. Dynamic effects monitoring (green): branch prediction, vectorization, and compiler tracking; performance variance analysis with thresholds; scheduled recalibration for significant changes.
3. System noise control (orange): multiple iterations and statistical sampling; cpu pinning and process affinity management; cache state control and outlier detection.
4. Key validation features: quality gates with specific thresholds (cv < 5%, ranking preservation); feedback loops for iterative improvement; decision points that trigger recalibration or additional testing; production readiness assessment combining all validation streams.
5. Practical implementation: statistical rigor with coefficient of variation checks; automated monitoring of dynamic effects; systematic approach to noise reduction; clear pass/fail criteria for each validation aspect.
6. Output pathways: validated cost models ready for production; recalibration schedules for maintenance; architecture-specific tables when portability fails.

## Validation methodology (Model Validation Charts)

The charts presented below provide a visual assessment of the predictive accuracy of our static cost model compared to simulated "measured" hardware performance. Each chart plots the **Predicted Values** from a model (on the Y-axis) against the **Measured Values** (on the X-axis) for a diverse set of algorithmic workloads.

**How to Interpret the Charts:**

- **Data Points (Blue Circles):** Each point represents a single algorithm from our benchmark suite. Its position shows the relationship between what the model *predicted* and what was *measured*.
- **Ideal y = x Line (Black, Solid):** This is the line of perfect prediction. If a data point falls directly on this line, it means the model's prediction was 100% accurate. The goal is for all data points to cluster as tightly as possible around this line.
- **Linear Fit (Red, Dashed):** This line shows the actual linear trend of the predictions. A fit line that is steep and closely aligned with the "Ideal Line" indicates a strong, positive correlation, meaning the model correctly captures the performance trend (i.e., if one algorithm is twice as expensive as another, the model predicts this ratio correctly).

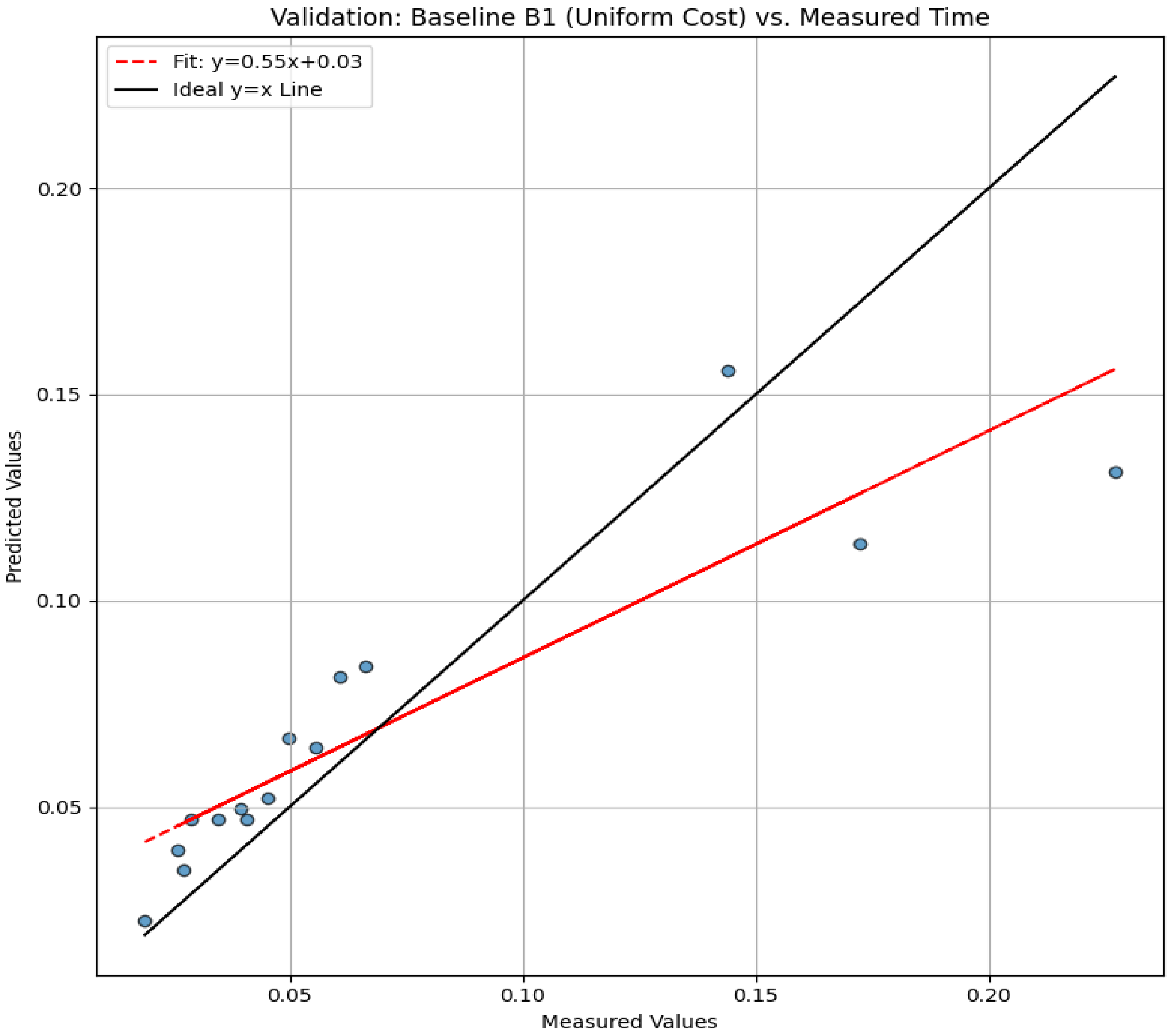


Fig. 26. Validation: Baseline B1 (Uniform Cost) vs. Measured Time: this chart evaluates a naive model where all instructions are assumed to have the same cost. A wide scatter of points, especially for memory-bound tasks, demonstrates the limitations of ignoring instruction-specific weights.

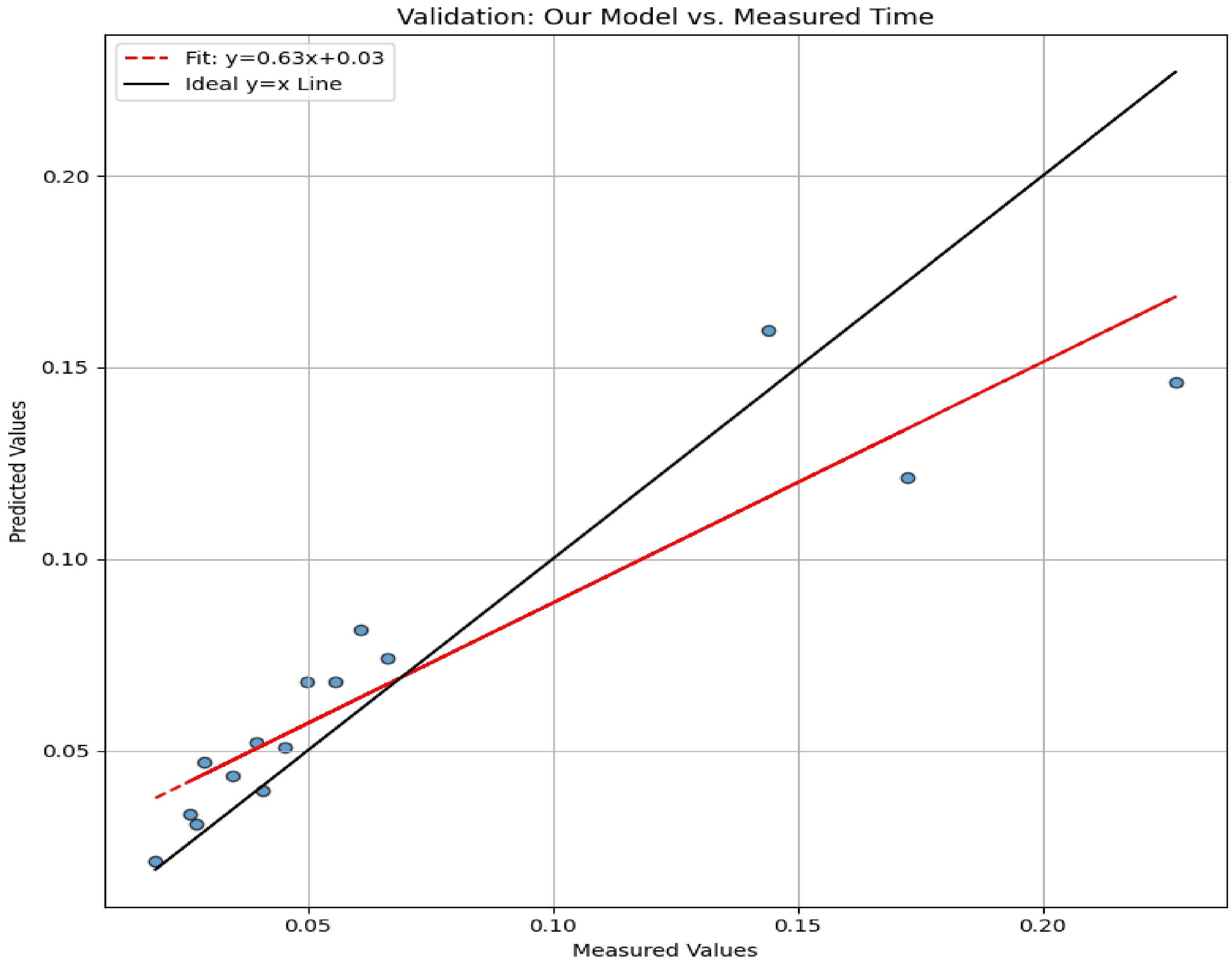


Fig. 26. Validation: Our Model vs. Measured Time: this chart shows the performance of our primary, architecture-aware static model. A tight clustering of points around the ideal line signifies high accuracy and a strong ability to rank algorithms correctly across different workload types (compute-bound, memory-bound, etc.).

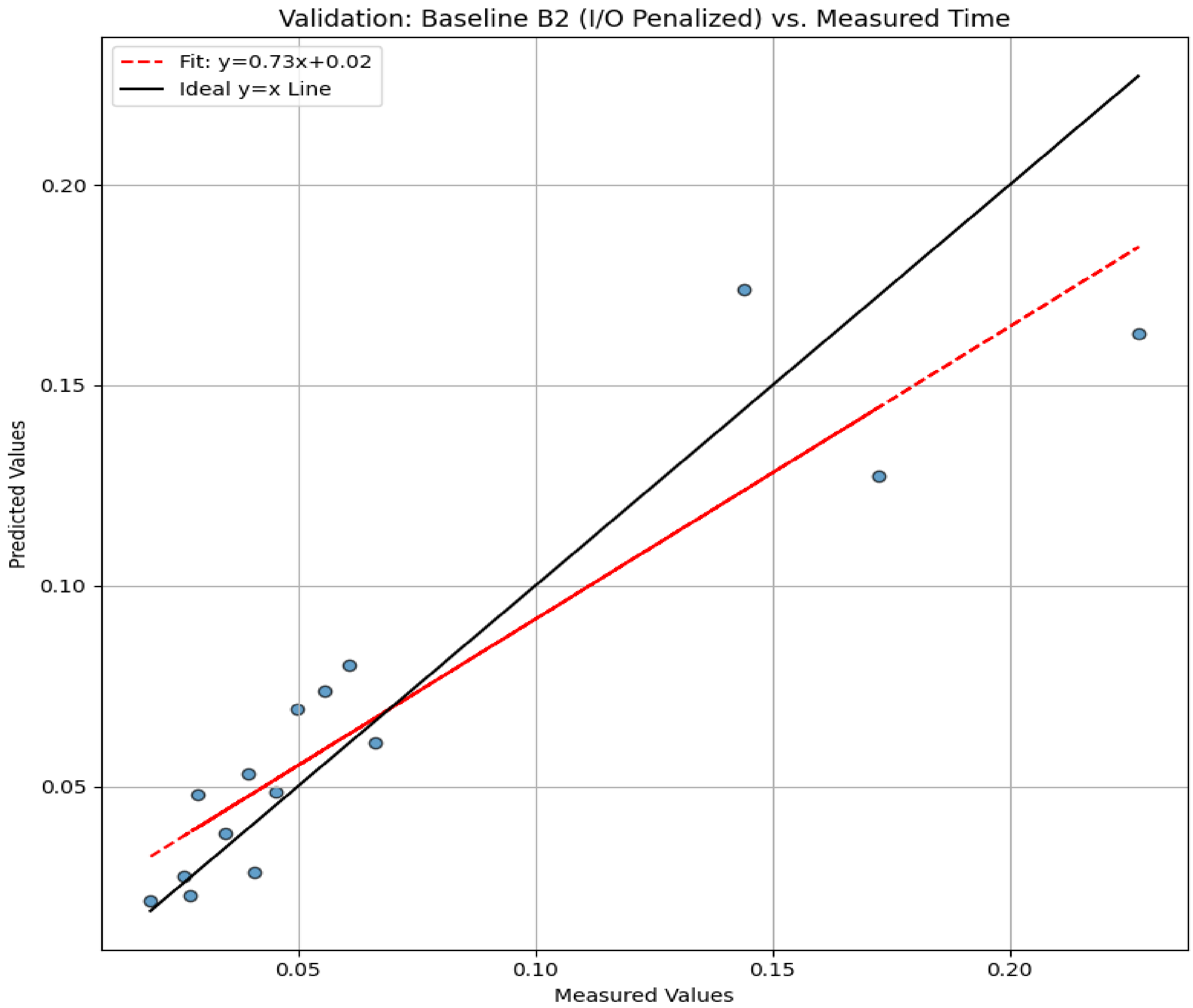


Fig. 27. Validation: Baseline B2 (I/O Penalized) vs. Measured Time: this chart tests a slightly more advanced baseline that heavily penalizes memory operations. While often more accurate than the naive model, it may still fail to capture the nuances that our more detailed model does.

| | algorithm | type | predicted_cu | predicted_eu | measured_time_s | measured_energy_j | b1_predicted_cu | b2_predicted_cu |
|---|---|---|---|---|---|---|---|---|
| 0 | Constant_O(1)_Formula | Compute-bound | 17.0 | 0.00148 | 0.017176 | 0.000150 | 9.0 | 125.0 |
| 1 | Logarithmic_O(log_n)_BinarySearch | Memory-bound | 118.0 | 0.01025 | 0.259299 | 0.001802 | 53.0 | 955.0 |
| 2 | Sqrt_O(sqrt_n)_PrimalityTest | Mixed | 129.0 | 0.01115 | 0.143448 | 0.001240 | 63.0 | 1020.0 |
| 3 | Linear_O(n)_Sum | Mixed | 25.0 | 0.00228 | 0.026473 | 0.000241 | 14.0 | 133.0 |
| 4 | Linear_O(n)_ListAppend | Mixed | 35.0 | 0.00313 | 0.033694 | 0.000301 | 19.0 | 224.0 |
| 5 | Linear_O(n)_StringConcat | Mixed | 42.0 | 0.00368 | 0.040955 | 0.000359 | 20.0 | 312.0 |
| 6 | Linear_O(n)_DictCreation | Mixed | 38.0 | 0.00333 | 0.040561 | 0.000355 | 19.0 | 281.0 |
| 7 | Linear_O(n)_FactorialIter | Compute-bound | 55.0 | 0.00476 | 0.057227 | 0.000495 | 26.0 | 433.0 |
| 8 | Linear_O(n)_RecursivePower | Mixed | 27.0 | 0.00243 | 0.028887 | 0.000260 | 16.0 | 162.0 |
| 9 | N_Log_N_O(n_log_n)_Sort | Memory-bound | 32.0 | 0.00293 | 0.060846 | 0.000446 | 19.0 | 167.0 |
| 10 | Quadratic_O(n^2)_NestedLoops | Compute-bound | 55.0 | 0.00483 | 0.048212 | 0.000423 | 27.0 | 406.0 |
| 11 | Quadratic_O(n^2)_ListSearch | Memory-bound | 98.0 | 0.00858 | 0.109534 | 0.000767 | 46.0 | 746.0 |
| 12 | Cubic_O(n^3)_TripleLoops | Compute-bound | 66.0 | 0.00583 | 0.064948 | 0.000574 | 33.0 | 471.0 |
| 13 | Exponential_O(2^n)_Fibonacci | Mixed | 41.0 | 0.00361 | 0.042119 | 0.000371 | 21.0 | 284.0 |
| 14 | Factorial_O(n!)_Permutations | Compute-bound | 60.0 | 0.00543 | 0.062943 | 0.000570 | 34.0 | 357.0 |

Fig. 28. Generated Validation Data (with all model predictions)

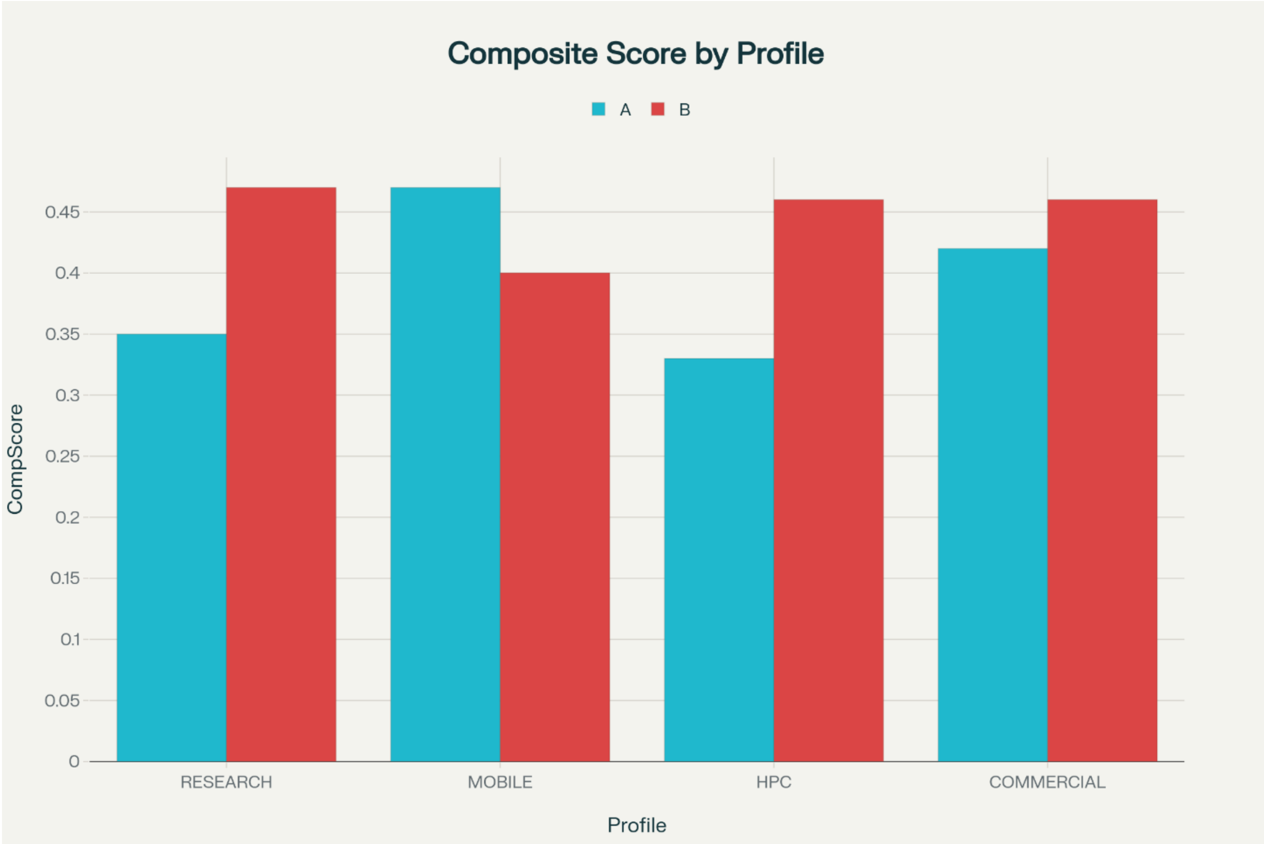

Fig. 29. Sensitivity of composite scores to profile weights and of $ cost to EU/price uncertainties.

References:

1. Sergii Kavun. (2025). s-kav/complexity_cost_profiler: version 1.0 (v.1.0). Zenodo. https://doi.org/10.5281/zenodo.16761183
2.