Empirical Comparison of Search Heuristics for Genetic Improvement of Software: Supplementary Material

Aymeric Blot and Justyna Petke

APPENDIX A
SEARCH APPROACHES PSEUDO CODES

Pseudo codes for search processes used for the experiments:

- random search (Algorithm 1)
- genetic programming (Algorithm 2)
- first improvement (Algorithm 3)
- best improvement (Algorithm 4)
- Tabu search (Algorithm 5)

The mutation procedure used in all approaches is given in Algorithm 6.

APPENDIX B
ADDITIONAL STATISTICS

Table I gives for each approach on each scenario the percentage of software variants generated for which fitness was strictly better that the original software during training. Because training fitness varies from fold to fold, Table II allows for a 5% decrease in fitness, therefore also including the percentage of rejected mutants that might have been useful using different instances.

Table III and Table IV detail, for every type of approach and across all repeated runs, the average and median number of software variants generated during the search, respectively. Local search approaches are separated according to the number of instances they use during training.

Table V reports number of valid mutated software found across all GI approaches, together with the percentage of syntactical uniqueness.

APPENDIX C
COMPLETE STATISTICAL ANALYSIS

Table VI and Table VII report the full statistical analysis at the end of the GI process (after the test step, using test data), including all 36 approaches. Table VI shows the Siegel post hoc analysis while Table VII shows the cliff delta effect size.

Table VIII and Table IX reports the full statistical analysis using all available instances, therefore also including training and validation.

APPENDIX D
SOURCE CODE DIFFS

Figure 1 to Figure 21 report for each software the specific patches inducing significant running time improvements. Improvements are reported using all available instances.

APPENDIX E
DETAILED PERFORMANCE

Figure 22 to Figure 31 report on individual instance performance for selected software variants.

A. Blot and J. Petke are with the Department of Computer Science, University College, London WC1E 6BT, U.K. (e-mail: a.blot@cs.ucl.ac.uk; j.petke@ucl.ac.uk).
This work was supported by UK EPSRC Fellowship EP/P023991/1.

Algorithm 1 Random Search

```
▷ m: maximal number of edits
procedure Rand(m)
    best ← empty mutant
    repeat
        ▷ Create a mutant at random with up to m edits
        mutant ← new mutant
        ▷ Accept if better
        if mutant < best then
            best ← mutant
        end if
    until training budget exhausted
    return best
end procedure
```

Algorithm 2 Genetic programming search

```
▷ n: population size
procedure GP(n)
    ▷ Initial generation, generated at random
    pop ← [ ]
    while |pop| < n do
        mutant ← new mutant
        append mutant to pop
    end while
    ▷ Subsequent generations
    repeat
        offspring ← [ ]
        ▷ (1) Selection (here: filter and sort)
        parents ← selection(pop)
        ▷ (2) Offspring by crossover
        for all parent1 ∈ parents[0 . . . n/2] do
            parent2 ← individual from parents (uniformly at random)
            mutant ← crossover(parent1, parent2) or crossover(parent2, parent1)
            append mutant to offspring
        end for
        ▷ (3) Offspring by mutation
        for all parent ∈ parents[0 . . . n/2] do
            mutant ← mutation(parent1)
            append mutant to offspring
        end for
        ▷ (4) If not enough parents: fill with random mutants
        while |offspring| < n do
            mutant ← new mutant
            append mutant to offspring
        end while
        pop ← offspring
    until training budget exhausted
    return Best mutant of the final generation
end procedure
```

Algorithm 3 First improvement local search

```
procedure First()
    best ← empty mutant
    repeat
        ▷ Append or remove an edit at random
        mutant ← mutate(best)
        ▷ Accept if better
        if mutant ≤ best then
            best ← mutant
        end if
    until training budget exhausted
    return best
end procedure
```

Algorithm 4 Best improvement local search

```
▷ s: neighbourhood size
procedure Best(s)
    best ← empty mutant
    repeat
        current ← best
        for i := 1 to s do
            ▷ Add or remove an edit at random
            mutant ← mutate(current)
            ▷ Accept if better
            if mutant ≤ best then
                best ← mutant
            end if
        end for
    until training budget exhausted
    return best
end procedure
```

Algorithm 5 Tabu Search

$\triangleright$ s: neighbourhood size
$\triangleright$ l: tabu list length
procedure $Tabu(s, l)$
  $best \leftarrow$ empty mutant
  $current \leftarrow$ empty mutant
  $tabu \leftarrow$ empty list
  repeat
    $local \leftarrow \varnothing$
    for $i := 1$ **to** s **do**
      $\triangleright$ Add or remove an edit at random
      $mutant \leftarrow mutate(current)$
      $\triangleright$ Abort if tabu
      if $mutant \notin tabu$ **then**
        next
      end if
      $\triangleright$ Check if better
      if $i = 1$ **or** $mutant \leq local$ **then**
        $local \leftarrow mutant$
      end if
    end for
    $current \leftarrow local$
    append $current$ to $tabu$
    discard oldest mutants if $|tabu| > l$
  until training budget exhausted
  return $best$
end procedure

Algorithm 6 Mutation

procedure $mutation(current)$
  $mutant \leftarrow current$
  $\triangleright$ Append or remove an edit at random
  if $|current| > 1$ **and** $rand() > 0.5$ **then**
    delete one edit from $mutant$
  else
    append one random edit to $mutant$
  end if
  return $mutant$
end procedure

TABLE I
PERCENTAGE OF SOFTWARE VARIANTS WITH FITNESS RATIO BETTER THAN 100% DURING TRAINING.

	MiniSAT		Sat4j	OptiPNG			MOEA	NSGA-II
Algo	CIT	Uniform	Uniform	Colour	Grey	Both	110%	
$Rand_2(1)$	5.1%	1.3%	33.9%	0.3%	0.4%	0.3%	5.6%	2.5%
$Rand_2(2)$	3.2%	1.0%	30.2%	0.2%	0.3%	0.3%	3.8%	2.4%
$Rand_2(5)$	1.7%	0.6%	31.3%	0.1%	0.1%	0.2%	1.9%	1.0%
$Rand_2(10)$	1.0%	0.3%	21.3%	0.1%	0.1%	0.1%	0.8%	0.6%
GP_1c	11.4%	17.8%	32.6%	**0.5**%	0.4%	0.4%	7.3%	4.6%
GP_11p	6.6%	7.1%	33.6%	0.3%	0.3%	0.3%	6.2%	3.8%
GP_1uc	7.4%	5.2%	33.2%	0.3%	0.4%	0.4%	6.6%	4.3%
GP_1ui	8.7%	8.1%	33.1%	0.3%	0.3%	0.3%	6.8%	4.2%
GP_1^rc	**11.5**%	**18.2**%	33.2%	**0.5**%	**0.7**%	**0.7**%	**7.4**%	5.6%
GP_1^r1p	7.6%	5.3%	31.1%	0.4%	0.6%	0.4%	6.3%	4.4%
GP_1^ruc	8.3%	5.1%	32.6%	0.5%	0.6%	0.4%	7.1%	4.9%
GP_1^rui	9.1%	8.7%	34.9%	0.4%	0.5%	0.4%	7.1%	5.7%
$First_1$	7.8%	5.9%	31.9%	0.1%	0.2%	0.4%	1.8%	5.3%
$Best_1$	8.4%	5.5%	33.3%	0.1%	0.2%	0.2%	2.1%	5.7%
$Tabu_1$	8.6%	5.7%	24.7%	0.1%	0.2%	0.3%	1.9%	5.8%
$First_2$	8.4%	6.5%	**36.4**%	0.2%	0.2%	0.3%	5.0%	**6.5**%
$Best_2$	10.5%	6.5%	31.7%	0.0%	0.1%	0.3%	4.6%	6.2%
$Tabu_2$	9.2%	7.2%	30.5%	0.3%	0.1%	0.2%	4.3%	5.8%

TABLE II
PERCENTAGE OF SOFTWARE VARIANTS WITH FITNESS RATIO BETTER THAN 105% DURING TRAINING.

	MiniSAT		Sat4j	OptiPNG			MOEA	NSGA-II
Algo	CIT	Uniform	Uniform	Colour	Grey	Both	110%	
$Rand_2(1)$	7.4%	3.2%	63.1%	0.6%	0.6%	0.8%	7.5%	5.4%
$Rand_2(2)$	5.3%	2.5%	62.4%	0.3%	0.4%	0.6%	5.4%	4.7%
$Rand_2(5)$	2.6%	1.3%	60.6%	0.2%	0.2%	0.3%	2.6%	2.3%
$Rand_2(10)$	1.6%	0.6%	51.2%	0.1%	0.1%	0.1%	1.1%	1.2%
GP_1c	14.9%	21.6%	57.8%	**0.8**%	0.8%	0.8%	9.3%	9.2%
GP_11p	9.6%	10.1%	56.0%	0.6%	0.5%	0.6%	7.7%	8.0%
GP_1uc	10.5%	7.8%	58.4%	0.5%	0.6%	0.7%	8.5%	8.3%
GP_1ui	12.3%	10.6%	59.5%	0.5%	0.5%	0.7%	8.9%	8.7%
GP_1^rc	**15.2**%	**21.9**%	60.6%	0.8%	**1.1**%	**1.0**%	9.4%	10.2%
GP_1^r1p	10.7%	8.1%	55.3%	0.6%	0.8%	0.7%	7.8%	8.7%
GP_1^ruc	11.1%	7.6%	58.4%	0.6%	0.9%	0.8%	8.8%	8.8%
GP_1^rui	12.6%	11.4%	61.8%	0.7%	0.8%	0.8%	**9.5**%	**10.4**%
$First_1$	8.7%	6.0%	54.5%	0.2%	0.3%	0.8%	2.3%	6.5%
$Best_1$	9.0%	5.7%	55.8%	0.2%	0.3%	0.6%	2.6%	7.0%
$Tabu_1$	9.2%	6.0%	53.6%	0.2%	0.3%	0.6%	2.4%	7.2%
$First_2$	9.8%	6.9%	**63.7**%	0.5%	0.4%	0.9%	6.4%	8.0%
$Best_2$	12.0%	6.8%	63.6%	0.2%	0.3%	0.9%	6.1%	8.4%
$Tabu_2$	10.8%	7.4%	62.6%	0.5%	0.3%	0.8%	5.9%	7.9%

TABLE III
AVERAGE NUMBER OF GENERATED SOFTWARE VARIANTS AND GENERATIONS DURING TRAINING.

	Random Search	Local Search		Genetic Programming	
Scenario	mutants (*2pb*)	mutants (*1pb*)	mutants (*2pb*)	mutants (*1pb*)	generations
MiniSAT (CIT)	544.8	339.4	168.7	262.7	3.2
MiniSAT (Uniform)	3848.5	2572.3	1418.5	1340.0	13.8
Sat4j (Uniform)	548.2	1029.3	527.1	858.3	9.0
OptiPNG (Colour)	9457.2	3933.9	2523.7	632.4	6.9
OptiPNG (Grey)	5377.1	2740.1	1770.9	613.7	6.7
Optipng (Both)	3324.9	1440.8	852.4	342.6	4.0
MOEA/D	280.1	1299.0	99.8	186.7	2.4
NSGA-II	460.5	252.0	135.3	225.8	2.9

1pb: approaches using a single instance per bin; *2pb*: approaches using two instances per bin

TABLE IV
MEDIAN NUMBER OF GENERATED SOFTWARE VARIANTS AND GENERATIONS DURING TRAINING.

	Random Search	Local Search		Genetic Programming	
Scenario	mutants (*2pb*)	mutants (*1pb*)	mutants (*2pb*)	mutants (*1pb*)	generations
MiniSAT (CIT)	454	354	170	257	3
MiniSAT (Uniform)	3118	2617	1497	1310	13
Sat4j (Uniform)	540	958	528	857	9
OptiPNG (Colour)	1464	932	414	616	7
OptiPNG (Grey)	1816	908	468	616	7
Optipng (Both)	728	369	210	333	4
MOEA/D	183	1550	107	196	2
NSGA-II	428	246	132	232	3

1pb: approaches using a single instance per bin; *2pb*: approaches using two instances per bin

TABLE V
NUMBER OF VALID (AND PERCENTAGE OF UNIQUE) FINAL MUTANTS.

Scenario	Training	Validation	Test	Overall
MiniSAT (CIT)	144 (88%)	126 (83%)	110 (82%)	48 (92%)
MiniSAT (uniform)	180 (89%)	176 (88%)	176 (88%)	175 (88%)
Sat4j (uniform)	173 (92%)	151 (87%)	151 (87%)	147 (87%)
OptiPNG (colour)	90 (97%)	89 (93%)	89 (93%)	89 (93%)
OptiPNG (grey)	90 (96%)	88 (94%)	88 (94%)	88 (94%)
OptiPNG (both)	178 (92%)	177 (85%)	170 (86%)	169 (86%)
MOEA/D (110%)	159 (81%)	159 (62%)	145 (59%)	143 (61%)
NSGA-II (110%)	161 (84%)	158 (69%)	140 (66%)	130 (63%)

TABLE VI
SIEGEL POST HOC STATISTICAL ANALYSIS, CPU, ALL TEST DATA. (FRIEDMAN TEST: $p = 1.9 \times 10^{-6}$)

Approach	$First_1$ (v)	$Best_2$ (v)	$Tabu_1$ (v)	$Best_1$ (v)	$Tabu_2$ (v)	$Rand_2(5)$	$Rand_2(2)$	$First_2$ (v)	$Rand_2(5)$ (v)	$GP_1^r c$ (v)	$Rand_2(2)$ (v)	$GP_1 c$ (v)	$Best_2$	$Rand_2(10)$	$GP_1^r c$	$Tabu_2$	$GP_1^r 1p$	$GP_1^r 1p$ (v)	$Rand_2(10)$ (v)	$First_2$	$Best_1$	$Rand_2(1)$	$GP_1^r ui$	$GP_1^r uc$	$GP_1^r ui$ (v)	$GP_1 ui$	$Tabu_1$	$First_1$	$GP_1^r uc$ (v)	$GP_1 ui$ (v)	$GP_1 uc$ (v)	$Rand_2(1)$ (v)	$GP_1 uc$	$GP_1 c$	$GP_1 1p$	$GP_1 1p$ (v)	Rank
$First_1$ (v)						·	·	*	*	*	*	*	*	**	**	**	**	**	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	13.11
$Best_2$ (v)														·	·	·	*	*	*	*	*	*	*	*	*	*	*	**	**	**	**	**	**	**	***	***	14.97
$Tabu_1$ (v)														·	·	·	·	*	*	*	*	*	*	*	*	*	*	*	*	*	**	**	**	**	***	***	15.29
$Best_1$ (v)																	·	·	*	*	*	*	*	*	*	*	*	*	*	*	**	**	**	**	**	***	15.57
$Tabu_2$ (v)																		·	*	*	*	*	*	*	*	*	*	*	*	*	**	**	**	**	**	***	15.65
$Rand_2(5)$	·																		·	·	·	·	·	·	·	·	·	·	·	*	*	*	*	*	**	***	16.27
$Rand_2(2)$	·																			·	·	·	·	·	·	·	·	·	·	·	*	*	*	*	**	***	16.36
$First_2$ (v)	*																													·	·	·	*	*	*	**	16.89
$Rand_2(5)$ (v)	*																														·	·	*	*	*	**	17.10
$GP_1^r c$ (v)	*																														·	·	*	*	*	**	17.23
$Rand_2(2)$ (v)	*																																·	·	*	**	17.49
$GP_1 c$ (v)	*																																·	·	*	**	17.58
$Best_2$	*																																·	·	·	**	17.80
$Rand_2(10)$	**	·	·																																	*	18.35
$GP_1^r c$	**	·	·																																	*	18.40
$Tabu_2$	**	·	·																																	*	18.48
$GP_1^r 1p$	**	*	·	·																																*	18.59
$GP_1^r 1p$ (v)	**	*	*	·	·																															*	18.94
$Rand_2(10)$ (v)	***	*	*	*	*	·																														·	19.36
$First_2$	***	*	*	*	*	·	·																													·	19.45
$Best_1$	***	*	*	*	*	·	·																													·	19.46
$Rand_2(1)$	***	*	*	*	*	·	·																													·	19.53
$GP_1^r ui$	***	*	*	*	*	·	·																													·	19.56
$GP_1^r uc$	***	*	*	*	*	·	·																														19.60
$GP_1^r ui$ (v)	***	*	*	*	*	·	·																														19.61
$GP_1 ui$	***	*	*	*	*	·	·																														19.63
$Tabu_1$	***	*	*	*	*	·	·																														19.69
$First_1$	***	**	*	*	*	·	·																														19.73
$GP_1^r uc$ (v)	***	**	*	*	*	·	·																														19.82
$GP_1 ui$ (v)	***	**	*	*	*	*	·	·																													19.92
$GP_1 uc$ (v)	***	**	**	**	**	*	*	·	·	·																											20.38
$Rand_2(1)$ (v)	***	**	**	**	**	*	*	·	·	·																											20.43
$GP_1 uc$	***	**	**	**	**	*	*	*	*	*	·	·	·																								20.85
$GP_1 c$	***	**	**	**	**	*	*	*	*	*	·	·	·																								20.95
$GP_1 1p$	***	***	***	**	**	**	**	*	*	*	*	*	·																								21.36
$GP_1 1p$ (v)	***	***	***	***	***	***	***	**	**	**	**	**	**	*	*	*	*	*	·	·	·	·	·														22.60

TABLE VII
CLIFF DELTA EFFECT SIZE, CPU, ALL TEST DATA. (FRIEDMAN TEST: $p = 1.9 \times 10^{-6}$)

Approach	$First_1$ (v)	$Best_2$ (v)	$Tabu_1$ (v)	$Best_1$ (v)	$Tabu_2$ (v)	$Rand_2(5)$	$Rand_2(2)$	$First_2$ (v)	$Rand_2(5)$ (v)	$GP_1^r c$ (v)	$Rand_2(2)$ (v)	$GP_1 c$ (v)	$Best_2$	$Rand_2(10)$	$GP_1^r c$	$Tabu_2$	$GP_1^r 1p$	$GP_1^r 1p$ (v)	$Rand_2(10)$ (v)	$First_2$	$Best_1$	$Rand_2(1)$	$GP_1^r ui$	$GP_1^r uc$	$GP_1^r ui$ (v)	$GP_1 ui$	$Tabu_1$	$First_1$	$GP_1^r uc$ (v)	$GP_1 ui$ (v)	$GP_1 uc$ (v)	$Rand_2(1)$ (v)	$GP_1 uc$	$GP_1 c$	$GP_1 1p$	$GP_1 1p$ (v)	Rank
$First_1$ (v)					·			·		·		·	·		·	·	·	·	·	·	·	·	·	·	·	·	*	*	·	·	·	·	·	*	·	*	13.11
$Best_2$ (v)						·						·			·	·				·	·	·	·	·	·	·	·	·	·	·	·	·	·	*	·	·	14.97
$Tabu_1$ (v)															·	·				·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	15.29
$Best_1$ (v)						·				·		·			·	·				·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	15.57
$Tabu_2$ (v)	·					·			·						·	·					·						·	·		·		·		·	·	·	15.65
$Rand_2(5)$		·		·	·			·		·	·	·	·		·	*	·	·	·	·	*	·	·	·	·	·	*	*	·	*	·	·	·	*	*	*	16.27
$Rand_2(2)$										·		·	·		·	·				·	·		·	·	·	·	·	·	·	·	·	·	·	*	·	*	16.36
$First_2$ (v)	·					·									·	·				·	·						·	·				·		·		·	16.89
$Rand_2(5)$ (v)					·					·		·	·		·	·	·	·		·	·	·	·	·	·	·	*	*	·	·	·	·	·	*	*	*	17.10
$GP_1^r c$ (v)	·			·		·	·		·												·						·	·						·		·	17.23
$Rand_2(2)$ (v)						·						·			·	·				·	·		·	·	·	·	·	·		·	·	·	·	·	·	·	17.49
$GP_1 c$ (v)	·	·		·		·	·		·		·																·	·						·		·	17.58
$Best_2$	·					·	·		·												·						·	·						·		·	17.80
$Rand_2(10)$															·	·				·	·		·	·	·	·	·	·		·	·	·	·	·	·	·	18.35
$GP_1^r c$	·	·	·	·	·	·	·	·	·		·			·														·									18.40
$Tabu_2$	·	·	·	·	·	*	·	·	·		·			·			·											·						·			18.48
$GP_1^r 1p$	·					·			·							·					·						·	·						·	·	·	18.59
$GP_1^r 1p$ (v)	·					·			·												·						·	·						·	·	·	18.94
$Rand_2(10)$ (v)	·					·															·						·	·						·		·	19.36
$First_2$	·	·	·	·		·	·	·	·		·			·													·	·						·			19.45
$Best_1$	·	·	·	·	·	*	·	·	·	·	·		·	·			·	·	·					·		·			·								19.46
$Rand_2(1)$	·	·	·	·		·			·																		·	·						·			19.53
$GP_1^r ui$	·	·	·	·		·	·		·		·			·														·						·			19.56
$GP_1^r uc$	·	·	·	·		·	·		·		·			·							·						·	·						·			19.60
$GP_1^r ui$ (v)	·	·	·	·		·	·		·		·			·													·	·						·			19.61
$GP_1 ui$	·	·		·		·	·		·		·			·							·						·	·						·		·	19.63
$Tabu_1$	*	·	·	·	·	*	·	·	*	·	·	·	·	·			·	·	·	·		·		·	·	·			·	·	·		·		·		19.69
$First_1$	*	·	·	·	·	*	·	·	*	·	·	·	·	·	·	·	·	·	·	·		·	·	·	·	·			·	·	·		·		·		19.73
$GP_1^r uc$ (v)	·	·	·	·		·	·		·												·						·	·						·		·	19.82
$GP_1 ui$ (v)	·	·	·	·	·	*	·		·		·			·													·	·						·			19.92
$GP_1 uc$ (v)	·	·	·	·		·	·		·		·			·													·	·						·			20.38
$Rand_2(1)$ (v)	·	·	·	·	·	·	·	·	·		·			·																							20.43
$GP_1 uc$	·	·	·	·		·	·		·		·			·													·	·						·			20.85
$GP_1 c$	*	*	·	·	·	*	*	·	*	·	·	·	·	·		·	·	·	·	·		·	·	·	·	·			·	·	·		·		·		20.95
$GP_1 1p$	·	·	·	·	·	*	·		*		·			·			·	·									·	·						·			21.36
$GP_1 1p$ (v)	*	·	·	·	·	*	*	·	*	·	·	·	·	·			·	·	·							·			·								22.60

TABLE VIII
SIEGEL POST HOC STATISTICAL ANALYSIS, CPU, ALL OVERALL DATA. (FRIEDMAN TEST: $p = 1.6 \times 10^{-9}$)

Approach	$Rand_2(5)$	$Best_2$ (v)	$Best_1$ (v)	$Tabu_2$ (v)	$Tabu_1$ (v)	$Rand_2(5)$ (v)	$First_1$ (v)	$Rand_2(2)$	$Rand_2(2)$ (v)	$First_2$ (v)	$Rand_2(10)$ (v)	$GP_1^r c$ (v)	$GP_1 c$ (v)	$Best_2$	$GP_1^r c$	$Rand_2(10)$	$GP_1 ui$ (v)	$GP_1^r ui$ (v)	$Tabu_2$	$GP_1 ui$	$GP_1^r ui$	$GP_1^r uc$ (v)	$Rand_2(1)$ (v)	$GP_1^r uc$	$GP_1^r 1p$	$Rand_2(1)$	$GP_1 uc$ (v)	$GP_1 uc$	$GP_1^r 1p$ (v)	$First_1$	$Best_1$	$First_2$	$GP_1 c$	$GP_1 1p$ (v)	$Tabu_1$	$GP_1 1p$	Rank
$Rand_2(5)$												·	·	·	·	*	*	*	*	*	**	**	**	**	**	**	**	***	***	***	***	***	***	***	***	***	14.45
$Best_2$ (v)															·	·	·	*	*	*	*	*	**	**	**	**	**	**	**	**	**	***	***	***	***	***	14.78
$Best_1$ (v)															·	·	·	*	*	*	*	*	**	**	**	**	**	**	**	**	**	***	***	***	***	***	14.79
$Tabu_2$ (v)																		·	·	·	*	*	*	*	*	*	**	**	**	**	**	**	**	***	***	***	15.45
$Tabu_1$ (v)																			·	·	*	*	*	*	*	*	*	**	**	**	**	**	**	**	***	***	15.69
$Rand_2(5)$ (v)																					·	·	*	*	*	*	*	**	**	**	**	**	**	**	***	***	15.86
$First_1$ (v)																					·	·	*	*	*	*	*	*	*	**	**	**	**	**	**	***	15.99
$Rand_2(2)$																					·	·	*	*	*	*	*	*	*	*	**	**	**	**	**	***	16.05
$Rand_2(2)$ (v)																							·	·	·	·	*	*	*	*	*	*	*	**	**	**	16.65
$First_2$ (v)																							·	·	·	·	*	*	*	*	*	*	*	**	**	**	16.66
$Rand_2(10)$ (v)																												·	·	·	·	·	·	*	*	**	17.41
$GP_1^r c$ (v)	·																											·	·	·	·	·	·	*	*	**	17.48
$GP_1 c$ (v)	·																											·	·	·	·	·	·	*	*	**	17.58
$Best_2$	·																													·	·	·	·	*	*	**	17.64
$GP_1^r c$	·	·	·																															*	*	*	17.97
$Rand_2(10)$	*	·	·																															*	*	*	18.07
$GP_1 ui$ (v)	*	·	·																															·	*	*	18.17
$GP_1^r ui$ (v)	*	*	*	·																														·	·	*	18.49
$Tabu_2$	*	*	*	·	·																														·	·	18.83
$GP_1 ui$	*	*	*	·	·																														·	·	18.84
$GP_1^r ui$	**	*	*	*	*	·	·	·																													19.38
$GP_1^r uc$ (v)	**	*	*	*	*	·	·	·																													19.43
$Rand_2(1)$ (v)	**	**	**	*	*	*	*	*	·	·																											19.81
$GP_1^r uc$	**	**	**	*	*	*	*	*	·	·																											19.82
$GP_1^r 1p$	**	**	**	*	*	*	*	*	·	·																											19.83
$Rand_2(1)$	**	**	**	*	*	*	*	*	·	·																											19.89
$GP_1 uc$ (v)	**	**	**	**	*	*	*	*	*	*																											20.39
$GP_1 uc$	***	**	**	**	**	**	*	*	*	*	·	·	·																								20.61
$GP_1^r 1p$ (v)	***	**	**	**	**	**	*	*	*	*	·	·	·																								20.65
$First_1$	***	**	**	**	**	**	**	*	*	*	·	·	·	·																							20.73
$Best_1$	***	**	**	**	**	**	**	**	*	*	·	·	·	·																							20.79
$First_2$	***	***	***	**	**	**	**	**	*	*	·	·	·	·																							20.87
$GP_1 c$	***	***	***	**	**	**	**	**	*	*	·	·	·	·																							20.89
$GP_1 1p$ (v)	***	***	***	***	**	**	**	**	**	**	*	*	*	*	*	*	·	·																			21.69
$Tabu_1$	***	***	***	***	***	***	**	**	**	**	*	*	*	*	*	*	*	·	·	·																	21.97
$GP_1 1p$	***	***	***	***	***	***	***	***	**	**	**	**	**	**	*	*	*	*	·	·																	22.39

TABLE IX

CLIFF DELTA EFFECT SIZE, CPU, ALL OVERALL DATA. (FRIEDMAN TEST: $p = 1.6 \times 10^{-9}$)

Approach	$Rand_2(5)$	$Best_2$ (v)	$Best_1$ (v)	$Tabu_2$ (v)	$Tabu_1$ (v)	$Rand_2(5)$ (v)	$First_1$ (v)	$Rand_2(2)$	$Rand_2(2)$ (v)	$First_2$ (v)	$Rand_2(10)$ (v)	$GP_1^r c$ (v)	$GP_1 c$ (v)	$Best_2$	$GP_1^r c$	$Rand_2(10)$	$GP_1 ui$ (v)	$GP_1^r ui$ (v)	$Tabu_2$	$GP_1 ui$	$GP_1^r ui$	$GP_1^r uc$ (v)	$Rand_2(1)$ (v)	$GP_1^r uc$	$GP_1^r 1p$	$Rand_2(1)$	$GP_1 uc$ (v)	$GP_1 uc$	$GP_1^r 1p$ (v)	$First_1$	$Best_1$	$First_2$	$GP_1 c$	$GP_1 1p$ (v)	$Tabu_1$	$GP_1 1p$	Rank
$Rand_2(5)$		·	·	·	·		·	·	·	·		·	·	·	·	·	·	·	*	·	*	·	·	*	·	·	*	*	·	*	*	*	*	*	*	*	14.45
$Best_2$ (v)	·											·	·	·	·		·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	*	·	*	·	14.78
$Best_1$ (v)	·					·						·	·	·	·				·	·	·		·	·	·	·	·	·	·	·	*	·	*	·	*	·	14.79
$Tabu_2$ (v)	·												·		·			·	·	·	·		·	·	·	·	·	·	·	·	·	·	·	·	*	·	15.45
$Tabu_1$ (v)	·													·	·				·	·	·		·	·	·	·	·	·	·	·	*	·	·	·	*	·	15.69
$Rand_2(5)$ (v)			·									·	·	·	·		·	·	*	·	*	·	·	·	·	·	·	·	·	*	*	*	*	*	*	*	15.86
$First_1$ (v)	·														·				·		·		·	·		·	·	·	·	·	·	·	·	·	*	·	15.99
$Rand_2(2)$	·												·	·	·				·	·	·			·			·	·		·	*	·	*	·	*	·	16.05
$Rand_2(2)$ (v)	·												·	·	·				·	·	·	·		·	·		·	·	·	·	*	·	*	·	*	·	16.65
$First_2$ (v)	·														·				·		·		·	·		·	·	·		·	·	·	·	·	*	·	16.66
$Rand_2(10)$ (v)													·	·	·			·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	*	·	*	·	17.41
$GP_1^r c$ (v)	·	·	·			·													·		·							·		·	·	·	·	·	·	·	17.48
$GP_1 c$ (v)	·	·	·	·		·		·	·		·																			·	·	·	·		·		17.58
$Best_2$	·	·	·		·	·		·	·		·																			·	·	·	·		·		17.64
$GP_1^r c$	·	·	·	·	·	·	·	·	·	·	·																			·	·	·	·		·		17.97
$Rand_2(10)$	·																		·		·			·		·	·	·	·	·	·	·	·	·	*	·	18.07
$GP_1 ui$ (v)	·	·				·													·		·			·		·		·		·	·	·	·	·	*	·	18.17
$GP_1^r ui$ (v)	·	·		·		·					·								·									·		·	·	·	·	·	·	·	18.49
$Tabu_2$	*	·	·	·	·	*	·	·	·	·	·	·				·	·	·				·								·	·	·	·		·		18.83
$GP_1 ui$	·	·	·	·	·	·		·	·		·																			·	·	·	·	·	·	·	18.84
$GP_1^r ui$	*	·	·	·	·	*	·	·	·	·	·	·				·	·																·		·		19.38
$GP_1^r uc$ (v)	·	·				·					·								·									·		·	·	·	·	·	*	·	19.43
$Rand_2(1)$ (v)	·	·	·	·	·	·	·			·	·																			·	·	·	·		·		19.81
$GP_1^r uc$	*	·	·	·	·	·	·	·	·	·	·					·	·													·	·		·		·		19.82
$GP_1^r 1p$	·	·	·	·	·	·			·		·																			·	·	·	·		·	·	19.83
$Rand_2(1)$	·	·	·	·	·	·	·			·	·					·	·														·		·		·		19.89
$GP_1 uc$ (v)	*	·	·	·	·	·	·	·	·	·	·					·														·	·	·	·		·		20.39
$GP_1 uc$	*	·	·	·	·	·	·	·	·	·	·	·				·	·	·				·								·	·	·	·		·		20.61
$GP_1^r 1p$ (v)	·	·	·	·	·	·	·		·		·					·														·	·	·	·		·		20.65
$First_1$	*	·	·	·	·	*	·	·	·	·	·	·	·	·	·	·	·	·	·	·		·	·	·	·		·	·	·					·		·	20.73
$Best_1$	*	·	*	·	*	*	·	*	*	·	·	·	·	·	·	·	·	·		·		·	·	·	·	·	·	·	·					·		·	20.79
$First_2$	*	·	·	·	·	*	·	·	·	·	·	·	·	·	·	·	·	·		·		·	·		·		·	·	·								20.87
$GP_1 c$	*	*	*	·	·	*	·	*	*	·	*	·	·	·	·	·	·	·		·	·	·	·	·	·	·	·	·	·								20.89
$GP_1 1p$ (v)	*	·	·	·	·	*	·	·	·	·	·	·				·	·	·		·		·								·	·				·		21.69
$Tabu_1$	*	*	*	*	*	*	*	*	*	*	*	·	·	·	·	*	*	·	·	·	·	*	·	·	·	·	·	·	·					·		·	21.97
$GP_1 1p$	*	·	·	·	·	*	·	·	·	·	·	·				·	·	·		·		·		·						·	·				·		22.39

```
*** before: core/Solver.cc
--- after: core/Solver.cc
***************
*** 399,404 ****
--- 399,405 ----
          for (int i = 1; i < c.size(); i++){
              Lit p  = c[i];
              if (!seen[var(p)] && level(var(p)) > 0){
+                 return false;
                  if (reason(var(p)) != CRef_Undef && (abstractLevel(var(p)) & abstract_levels) != 0){
                      seen[var(p)] = 1;
                      analyze_stack.push(p);
***************
*** 709,714 ****
--- 710,716 ----
              if (nof_conflicts >= 0 && conflictC >= nof_conflicts || !withinBudget()){
                  // Reached bound on number of conflicts:
                  progress_estimate = progressEstimate();
+                 learntsize_adjust_cnt     = (int)learntsize_adjust_confl;
                  cancelUntil(0);
                  return l_Undef; }
```

Fig. 1. MiniSAT (CIT) variant; overall ratio: 77.89%

```
*** before: core/Solver.cc
--- after: core/Solver.cc
***************
*** 792,797 ****
--- 792,798 ----
  }

      while (size-1 != x){
+         return false;
          size = (size-1)>>1;
          seq--;
          x = x % size;
```

Fig. 2. MiniSAT (uniform) variant; overall ratio: 32.2%

```
*** before: core/Solver.cc
--- after: core/Solver.cc
***************
*** 670,674 ****
          CRef confl = propagate();
          if (confl != CRef_Undef){
              // CONFLICT
!             conflicts++; conflictC++;
              if (decisionLevel() == 0) return l_False;
--- 670,674 ----
          CRef confl = propagate();
          if (confl != CRef_Undef){
              // CONFLICT
!             conflicts++;
              if (decisionLevel() == 0) return l_False;
```

Fig. 3. MiniSAT (uniform) variant; overall ratio: 38.6%

```
*** before: core/Solver.cc
--- after: core/Solver.cc
***************
*** 707,716 ****
--- 707,712 ----
          }else{
              // NO CONFLICT
-             if (nof_conflicts >= 0 && conflictC >= nof_conflicts || !withinBudget()){
-                 // Reached bound on number of conflicts:
-                 progress_estimate = progressEstimate();
-                 cancelUntil(0);
-                 return l_Undef; }

              // Simplify the set of problem clauses:
              if (decisionLevel() == 0 && !simplify())
```

Fig. 4. MiniSAT (uniform) variant; overall ratio: 38.6%

```
*** before: core/Solver.cc
--- after: core/Solver.cc
***************
*** 709,715 ****
--- 709,714 ----
              if (nof_conflicts >= 0 && conflictC >= nof_conflicts || !withinBudget()){
                  // Reached bound on number of conflicts:
                  progress_estimate = progressEstimate();
-                 cancelUntil(0);
                  return l_Undef; }

              // Simplify the set of problem clauses:
```

Fig. 5. MiniSAT (uniform) variant; overall ratio: 38.6%

```
*** before: org.sat4j.core/src/main/java/org/sat4j/minisat/core/Solver.java
--- after: org.sat4j.core/src/main/java/org/sat4j/minisat/core/Solver.java
***************
*** 1129,1140 ****
--- 1129,1134 ----
          constr.assertConstraint(this);
          int p = toDimacs(constr.get(0));
          this.slistener.adding(p);
-         if (constr.size() == 1) {
-             this.stats.incLearnedliterals();
-             this.slistener.learnUnit(p);
-         } else {
-             this.learner.learns(constr);
-         }
      }

      /**
```

Fig. 6. Sat4j variant; overall ratio: 83.6%

```
*** before: org.sat4j.core/src/main/java/org/sat4j/minisat/core/Solver.java
--- after: org.sat4j.core/src/main/java/org/sat4j/minisat/core/Solver.java
***************
*** 1133,1139 ****
              this.stats.incLearnedliterals();
              this.slistener.learnUnit(p);
          } else {
!             this.learner.learns(constr);
          }
      }
--- 1133,1142 ----
              this.stats.incLearnedliterals();
              this.slistener.learnUnit(p);
          } else {
!             if (this.timer != null) {
!                 this.timer.cancel();
!                 this.timer = null;
!             }
          }
      }
```

Fig. 7. Sat4j variant; overall ratio: 84.0%

```
*** before: src/optipng/optim.c
--- after: src/optipng/optim.c
***************
*** 1164,1185 ****
--- 1164,1170 ----
          png_set_compression_mem_level(write_ptr, memory_level);
          png_set_compression_strategy(write_ptr, compression_strategy);
          png_set_filter(write_ptr, PNG_FILTER_TYPE_BASE, filter_table[filter]);
-         if (compression_strategy != Z_HUFFMAN_ONLY &&
-             compression_strategy != Z_RLE)
-         {
-             if (options.window_bits > 0)
-                 png_set_compression_window_bits(write_ptr,
-                                                 options.window_bits);
-         }
-         else
-         {
- #ifdef WBITS_8_OK
-             png_set_compression_window_bits(write_ptr, 8);
- #else
-             png_set_compression_window_bits(write_ptr, 9);
- #endif
-         }

          /* Override the default libpng settings. */
          png_set_keep_unknown_chunks(write_ptr,
```

Fig. 8. OptiPNG variant; overall ratio: 57.4% (colour), 61.4% (gray), 58.9% (both)

```
*** before: src/optipng/optim.c
--- after: src/optipng/optim.c
***************
*** 1177,1183 ****
  #ifdef WBITS_8_OK
              png_set_compression_window_bits(write_ptr, 8);
  #else
!             png_set_compression_window_bits(write_ptr, 9);
  #endif
          }

--- 1177,1183 ----
  #ifdef WBITS_8_OK
              png_set_compression_window_bits(write_ptr, 8);
  #else
!             const char *con_str, *log_str;
  #endif
          }
```

Fig. 9. OptiPNG variant; overall ratio: 57.4% (colour), 61.4% (gray)

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 328,334 ****
--- 328,334 ----
          // calculate igd-values
          if(gen%dd==0)
          {
-             calc_distance();
              cout<<"gen = "<<gen<<"  gd = "<<distance<<"  "<<endl;
              gd.push_back(int(1.0*gen/dd)); gd.push_back(distance);
          }
```

Fig. 10. MOEA/D variant; overall ratio: 90.2%

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 326,337 ****
--- 326,332 ----
          int dd = int(max_gen/25.0);

          // calculate igd-values
-         if(gen%dd==0)
-         {
-             calc_distance();
-             cout<<"gen = "<<gen<<"  gd = "<<distance<<"  "<<endl;
-             gd.push_back(int(1.0*gen/dd)); gd.push_back(distance);
-         }

          // save the final population - F space
          if(gen%max_gen==0)
```

Fig. 11. MOEA/D variant; overall ratio: 90.6%

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 305,311 ****
--- 305,311 ----

      // load the representative Pareto-optimal solutions
      sprintf(filename,"PF/pf_%s.dat",strTestInstance);
-     loadpfront(filename,ps);

      // initialization
      int gen   = 1;
```

Fig. 12. MOEA/D variant; overall ratio: 90.2%

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 304,310 ****
--- 304,310 ----
      char filename[1024];

      // load the representative Pareto-optimal solutions
-     sprintf(filename,"PF/pf_%s.dat",strTestInstance);
      loadpfront(filename,ps);

      // initialization
```

Fig. 13. MOEA/D variant; overall ratio: 90.2%

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 390,400 ****
--- 390,395 ----
      for(int i=0; i<ps.size(); i++)
      {
          double min_d = 1.0e+10;
-         for(int j=0; j<population.size(); j++)
-         {
-             double d = dist_vector(ps[i].y_obj, population[j].indiv.y_obj);
-             if(d<min_d)  {min_d = d;}
-         }
          distance+= min_d;
      }
      distance/=ps.size();
```

Fig. 14. MOEA/D variant; overall ratio: 90.2%

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 207,212 ****
--- 207,213 ----
  void CMOEAD::update_reference(CMOEADInd &ind)
  {
      //ind: child solution
+     ps.clear();
      for(int n=0; n<nobj; n++)
      {
          if(ind.y_obj[n]<idealpoint[n])
```

Fig. 15. MOEA/D variant; overall ratio: 90.3%

```
*** before: DMOEA/dmoeafunc.h
--- after: DMOEA/dmoeafunc.h
***************
*** 174,178 ****
      int size, time = 0;
      if(type==1) {size = population[id].table.size();}
!     else        {size = population.size();}
      int *perm = new int[size];
      random_permutation(perm, size);
--- 174,178 ----
      int size, time = 0;
      if(type==1) {size = population[id].table.size();}
!     else        {}
      int *perm = new int[size];
      random_permutation(perm, size);
```

Fig. 16. MOEA/D variant; overall ratio: 64.8% (fitness: 131.0%)

```
*** before: NSGA2/nsga2func.h.xml
--- after: NSGA2/nsga2func.h.xml
***************
*** 142,153 ****
--- 142,147 ----
                      if(offspring[j]<offspring[k]&&!(offspring[j]==offspring[k])) {rank[k]++;}
-                     if(offspring[k]<offspring[j]&&!(offspring[j]==offspring[k]))
-                     {
-                         offspring[k].count++;
-                         int m = offspring[k].count - 1;
-                         cset[k][m] = j;
-                     }
                  }
              }
          }
***************
*** 216,221 ****
--- 211,215 ----
              {population.push_back(offspring[i]);}
          }
-         rank++;
          if(population.size()>=pops) {break;}
      }
***************
*** 278,284 ****
--- 273,278 ----
      char filename[1024]
-     sprintf(filename,"PF/pf_%s.dat",strTestInstance);
      loadpfront(filename,ps);
      nfes    = 0;
```

Fig. 17. NSGA-II variant; overall ratio: 50.6% (fitness: 100.2%)

```
*** before: NSGA2/nsga2func.h.xml
--- after: NSGA2/nsga2func.h.xml
***************
*** 361,371 ****
--- 361,366 ----
        for(int i=0; i<ps.size(); i++)
        {
            double min_d = 1.0e+10;
-           for(int j=0; j<population.size(); j++)
-           {
-               double d = dist_vector(ps[i].y_obj, population[j].y_obj);
-               if(d<min_d)  {min_d = d;}
-           }
            distance+= min_d;
        }
        distance/=ps.size();
```

Fig. 18. NSGA-II variant; overall ratio: 98.6% (fitness: no change)

```
*** before: NSGA2/nsga2func.h
--- after: NSGA2/nsga2func.h
***************
*** 216,220 ****
--- 216,219 ----
              {population.push_back(offspring[i]);}
          }
-         rank++;
          if(population.size()>=pops) {break;}
      }
```

Fig. 19. NSGA-II variant; overall ratio: 86.7% (fitness: 102.6%)

```
*** before: NSGA2/nsga2func.h
--- after: NSGA2/nsga2func.h
***************
*** 216,220 ****
--- 216,220 ----
              population.push_back(offspring[i]);
-         rank++;
          if(population.size()>=pops) {break;}
      }
***************
*** 278,284 ****
--- 278,284 ----

      char filename[1024];

-     sprintf(filename,"PF/pf_%s.dat",strTestInstance);
      loadpfront(filename,ps);
      nfes    = 0;
```

Fig. 20. NSGA-II variant; overall ratio: 85.3% (fitness: 102.6%)

```
*** before: NSGA2/nsga2func.h
--- after: NSGA2/nsga2func.h
***************
*** 81,89 ****
--- 81,86 ----
      bool flag = true;
      int  size = offspring.size();
      for(int i=0; i<size; i++){
-         if(ind==offspring[i]){
-             flag = false;
-             break;
-         }
      }
      if(flag) offspring.push_back(ind);
```

Fig. 21. NSGA-II variant; overall ratio: 89.1% (fitness: no change)

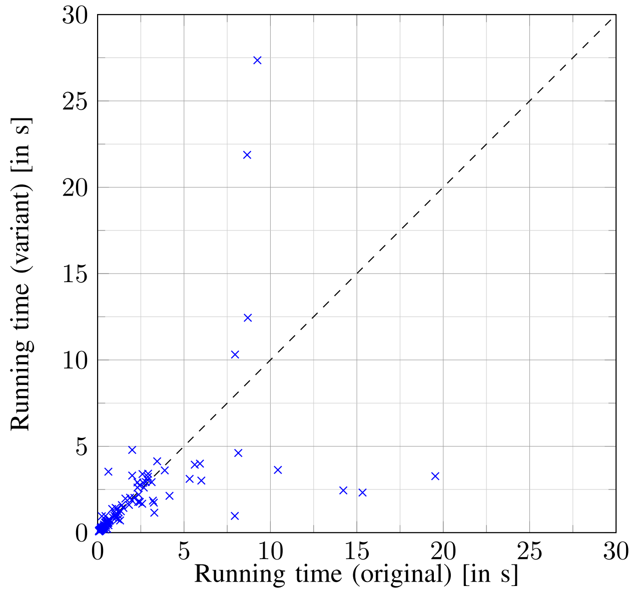

Fig. 22. All instance performance, MiniSAT (CIT) variant with overall ratio 77.89%

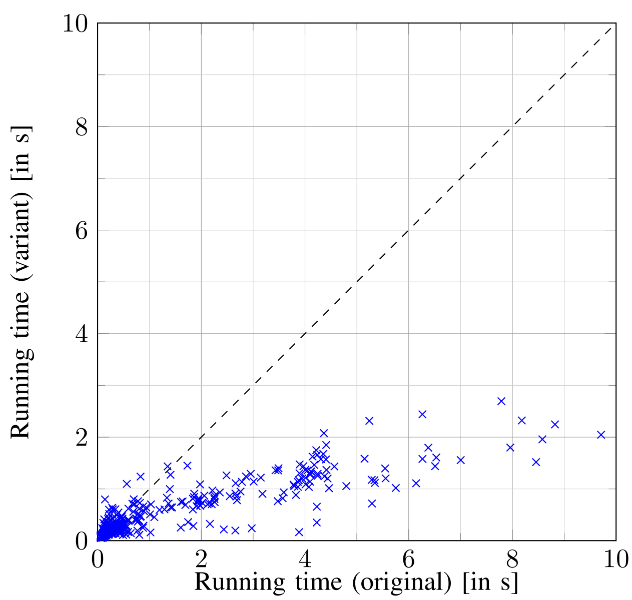

Fig. 23. All instance performance, MiniSAT (uniform) variant with overall ratio 32.2%

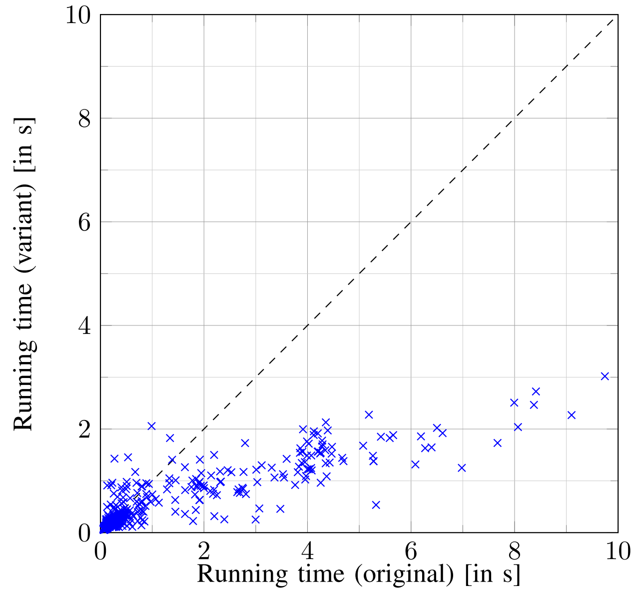

Fig. 24. All instance performance, MiniSAT (uniform) variant with overall ratio 38.6% ((`conflictc++` deletion))

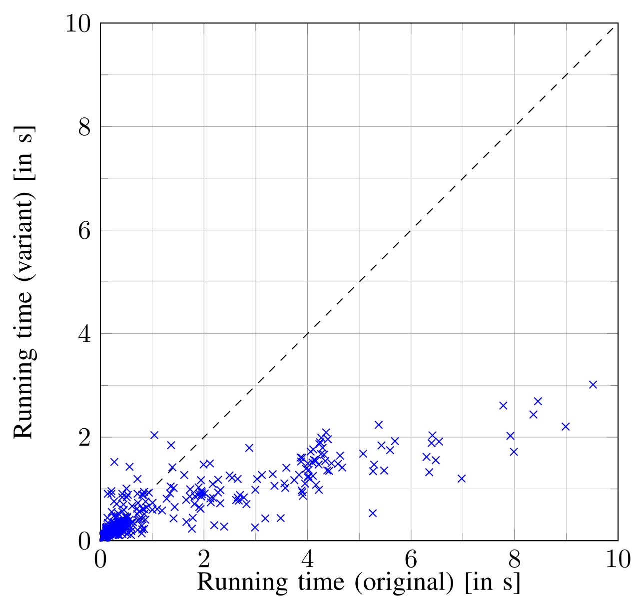

Fig. 25. All instance performance, MiniSAT (uniform) variant with overall ratio 38.6% (`cancelUntil(0)` deletion)

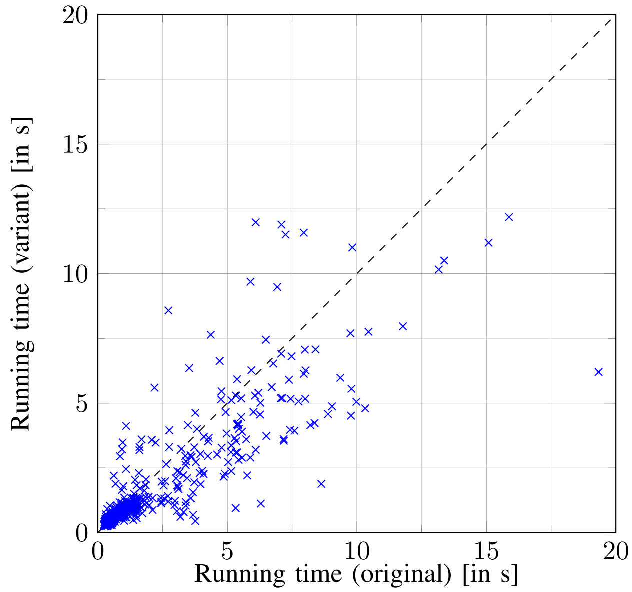

Fig. 26. All instance performance, Sat4j (uniform) variant with overall ratio 82.64%

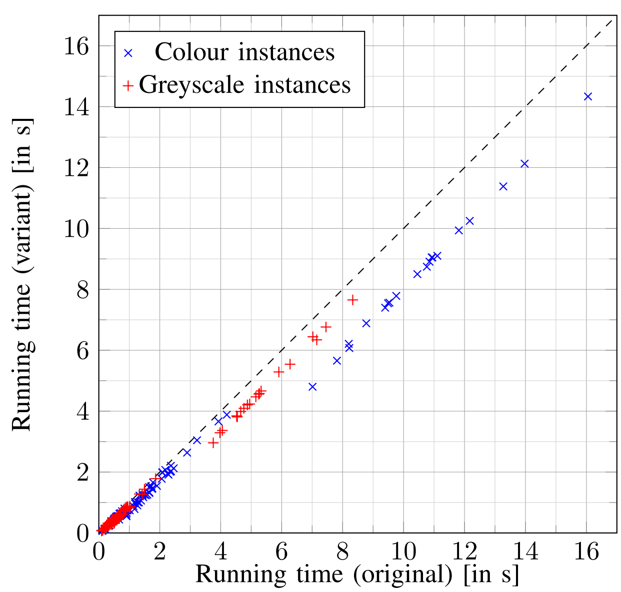

Fig. 27. All instance performance, OptiPNG variant with overall ratio 58.9%

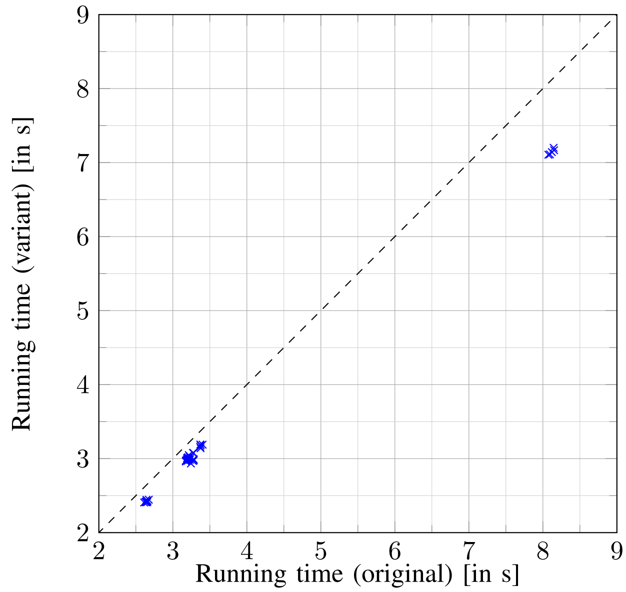

Fig. 28. All instance performance, MOEA/D variant with overall ratio 90.2% and no solution fitness change (`loadpfront(filename,ps)` deletion)

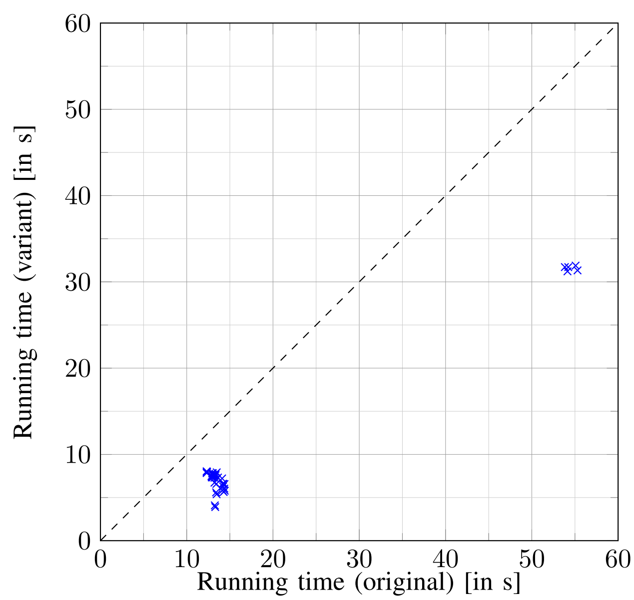

Fig. 29. All instance performance, NSGA-II variant with overall ratio 50.6% and 0.2% solution fitness loss

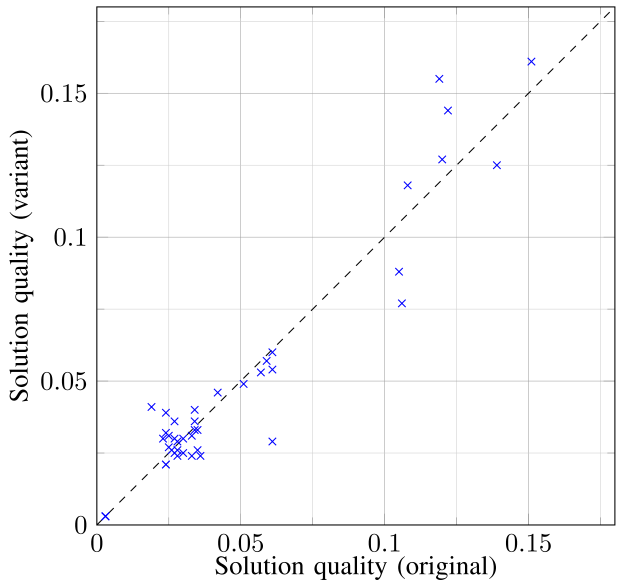

Fig. 30. All instance performance, NSGA-II variant with overall ratio 50.6% and 0.2% solution fitness loss

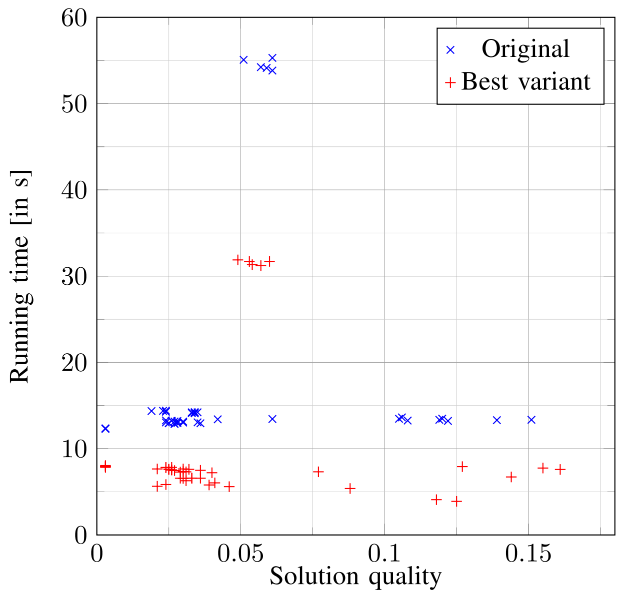

Fig. 31. All instance performance, NSGA-II variant with overall ratio 50.6% and 0.2% solution fitness loss