

# Adaptive Online Time Allocation to Search Algorithms

*Matteo Gagliolo, Viktor Zhumatiy, Jürgen Schmidhuber*

`{matteo,viktor,juergen}@idsia.ch`

*IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale*

Lugano, Switzerland

[www.idsia.ch](http://www.idsia.ch)

# Roadmap

## Framework

- What do we mean by AOTA
- Why/when would that be useful
- Previous/related work

## Experiments

- Our example AOTA
- Another example AOTA: Parameter-less GA
- Comparative experiments with GAs

# An example AOTA: the Student

A PhD student wants to study some new algorithm.

He first chooses a set of variants and/or different parameterizations of the algorithm, and some benchmark problem.

He then runs the chosen algorithms in parallel on his machine, checking their outputs (e.g. their *learning curves*) from time to time.

He could then *adjust the priorities* of the running algorithms *online* according to their performance.

He will soon realize that this task is so dull that even a machine can do it..

### Consider:

- a set  $R$  of problems  $r_1, r_2, \dots$
- a set  $A$  of iterative algorithms  $a_1, a_2, \dots, a_N$ , that can be
  - applied to the solution of the problems in  $R$
  - paused and resumed at any time, at a negligible cost
  - queried, at a negligible cost, for state information  $d \in R^D$  related to their current progress in solving  $r$

Typical approach: trial and error . . .

Meta-learning: learn to select a *single element* or a *subset* of  $A$  (to be executed in parallel, or in a given order)

More general: learn to *allocate time* to elements in  $A$

# Adaptive time allocation

Time can be allocated:

- **offline**: a fixed execution schedule is chosen before starting the selected algorithm(s)
- **online**: time is allocated to the  $a_i \in A$  *dynamically*, according to the evolution of their states  $d_i$

Adaptation can take place while solving:

- a single problem (*intra-problem* adaptation)
- a *sequence* of problems (*inter-problem* adaptation, or *inductive transfer*)

# Why/when AOTA ?

Computation time is often neglected in meta-learning

- some algorithms (esp. stochastic) can exhibit large performance fluctuations
- potentially large computational advantage over parallel execution
- no training data is available for new algorithms
- time to collect training data can be huge
- new problems might not be well represented in training set

*In all these cases we should have a feedback on actual performance* of the  $a$ 's

# Adaptive Online Time Allocator (AOTA)

At step  $k$  it tries to solve current problem  $r(k)$  by allocating computation time  $t(k)$  to algorithm  $a(k) \in A$ .

The pair

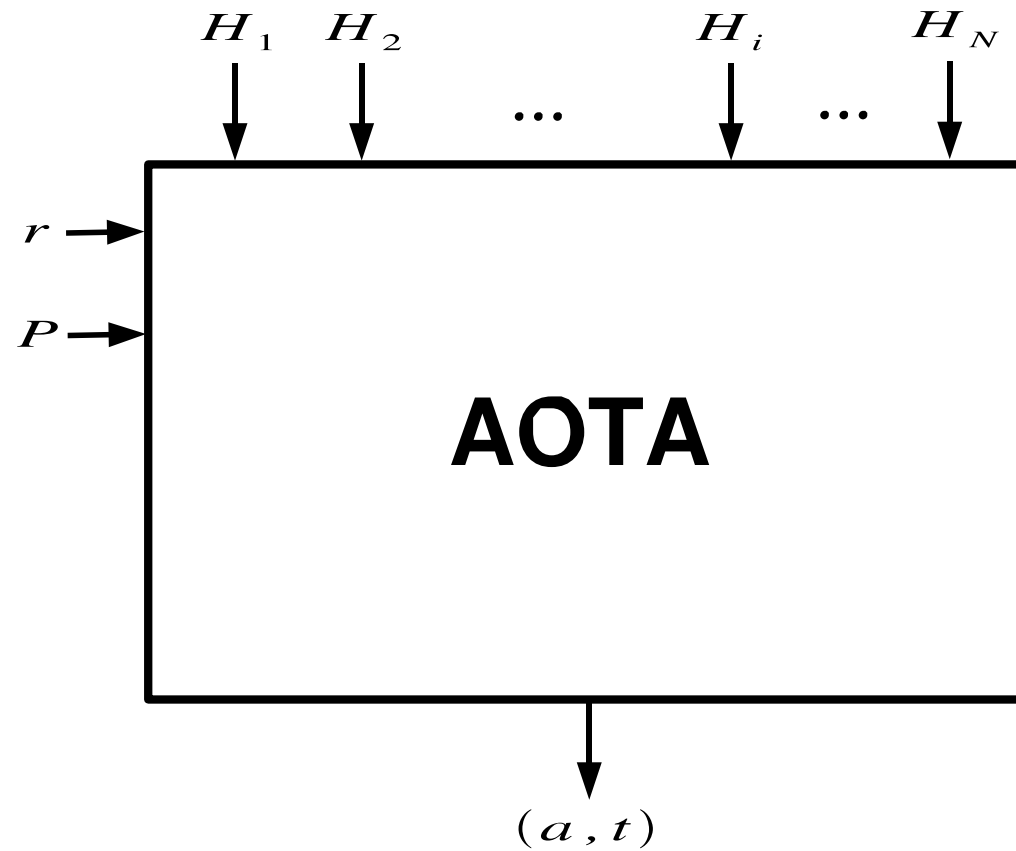
$$(a(k), t(k)) = f(r(k), H(k), P)$$

is generated based on the *history*

$$H(k) = \{(j, r(j), a(j), t(j), d(j)) : 0 < j < k\}$$

and the initial bias  $P$ .

# Adaptive Online Time Allocator (AOTA)



$N$  algorithms  $a_i$ , state history  $H_i$ , problem features  $r$ , initial bias  $P$ .

# Previous/Related work

Offline, inter-problem adaptation:

- Algorithm selection (Rice '76)
- Algorithm recommendation (Soares, ML '04)
- Algorithm portfolios (Gomes & Selman, AI '01)
- Anytime algorithms scheduling (Boddy & Dean, AI '94)
- Time-limited planning (Horvitz & Zilberstein, AI '01; Russel & Wefald, AI '91)
- Optimal Ordered Problem Solver (Schmidhuber, ML '04)
- Racing algorithms (Moore & Lee, ICML '94; Birattari et al., GECCO '02)

## Previous/Related work

### Limited online adaptation:

- Incremental Machine Learning (Solomonoff, IDSIA techrep '03)
- Parameterless GA (Harick & Lobo, GECCO '99)
- Algorithm Selection using RL (Lagoudakis & Littman, ICML '00)
- Bayesian approach (Horvitz et al., UAI '01)
- Anytime algorithm monitoring (Hansen & Zilberstein, AI '01)

## Previous/Related work

### Time allocation

online

offline

intra

- Our AOTA
- Parameterless GA

- A-priori choice

inter

- Solomonoff's incremental ML
- Algorithm Selection using RL
- Bayesian approach
- Anytime alg. monitoring

- Algorithm selection
- Algorithm portfolios
- Anytime alg. scheduling
- Time-limited planning
- OOPS
- Racing

# Objectives

- a generic intra-problem AOTA that
  - performs as good as Parameter-less GA
  - can be extended to other parameters (see below)
- a feasibility study for inter-problem AOTA

# Our AOTA

Algorithm set  $A = \{a_i, i = 1..N\}$ , bias  $P_A(k) = \{p_i(k), i = 1..N\}$ .

Machine time sliced in small  $\Delta T$ .

While ( $r$  not solved)

    Update  $P_A(k)$

    For each  $i = 1..N$

        Generate and execute pair  $(a(k), t(k)) = (a_i, p_i \Delta T)$

$k = k + 1$

    End

End

## Our AOTA

Algorithm set  $A = \{a_i, i = 1..N\}$ , bias  $P_A(k) = \{p_i(k), i = 1..N\}$ , algorithm state histories  $H_i(k) = \{(t(k), d(k)) | a(k) = a_i\}$ , algorithm utility values  $U_A(k) = \{u_i(k), i = 1..N\}$ .

Set  $\Delta T$ , initialize  $u_i(0), i = 1..N$ , and set  $k = 1$ .

While ( $r$  not solved)

Update  $P_A(k) = f_P(U_A(k))$

For each  $i = 1..N$

Generate and execute pair  $(a(k), t(k)) = (a_i, p_i \Delta T)$

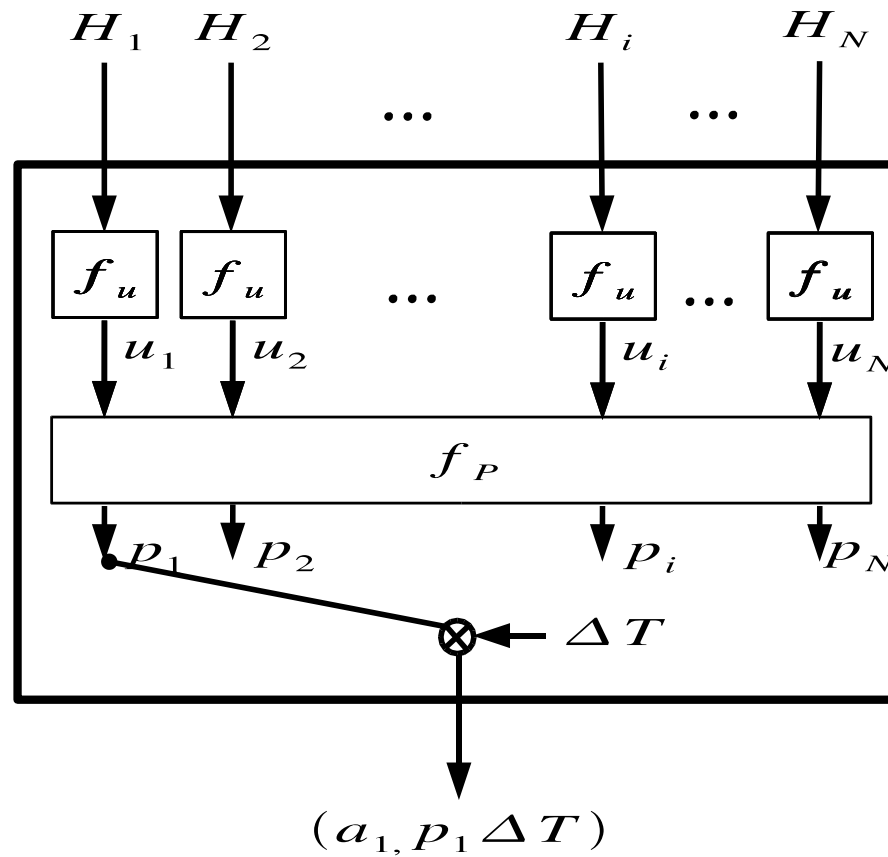
Update  $u_i(k + 1) = f_u(H_i(k))$ .

$k = k + 1$

End

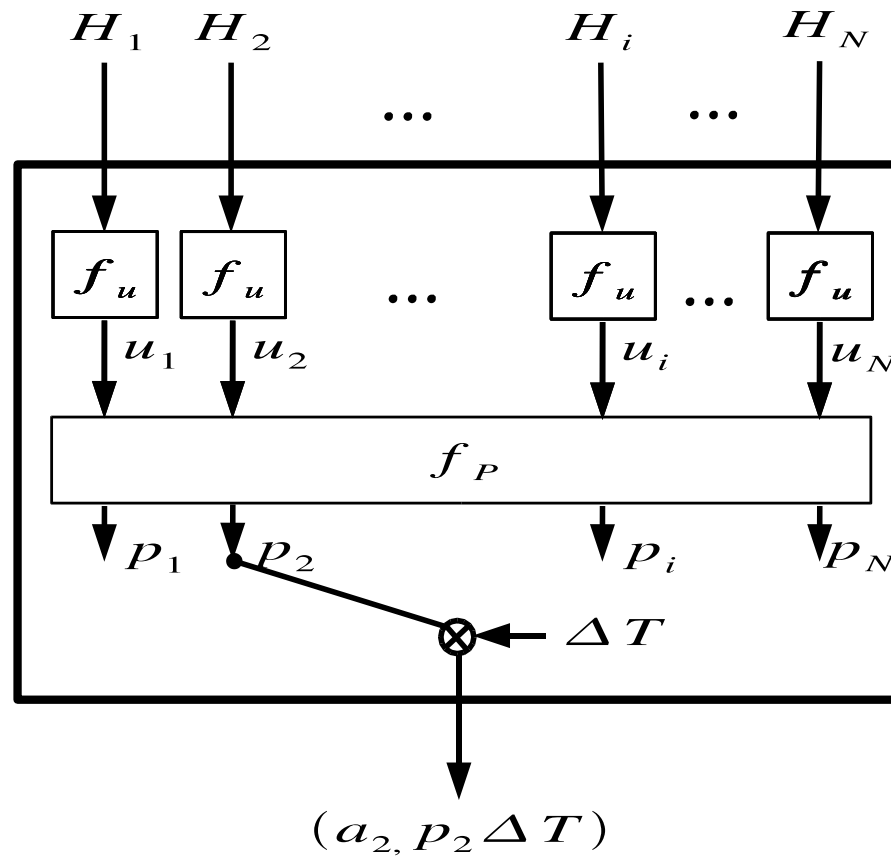
End

# Our AOTA



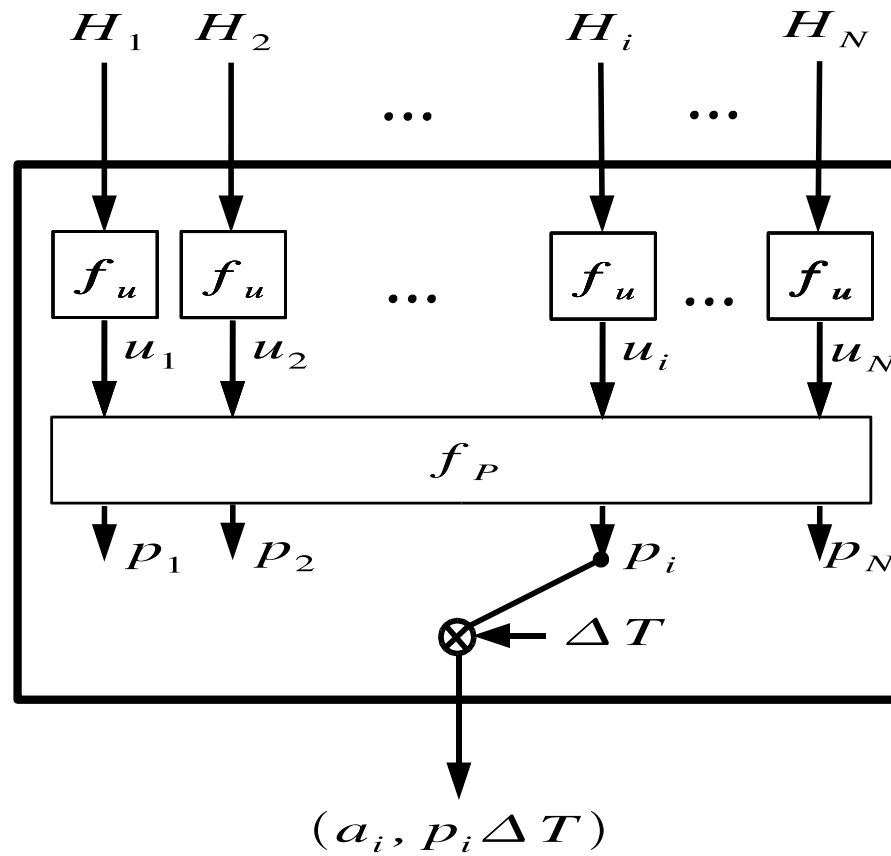
$N$  algorithms  $a_i$ , bias  $p_i$ , state history  $H_i$ , utility value  $u_i$ .

# Our AOTA



$N$  algorithms  $a_i$ , bias  $p_i$ , state history  $H_i$ , utility value  $u_i$ .

# Our AOTA



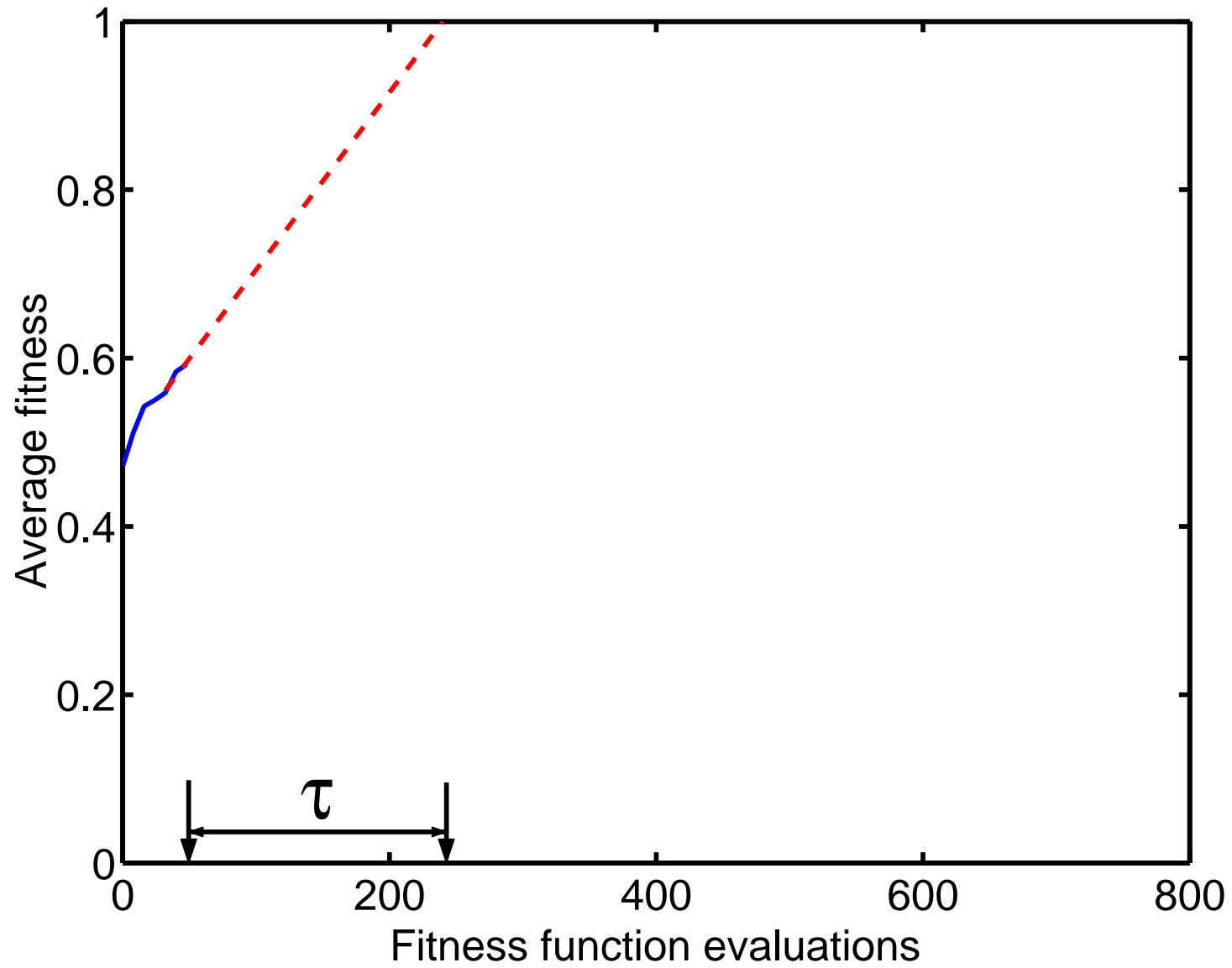
$N$  algorithms  $a_i$ , bias  $p_i$ , state history  $H_i$ , utility value  $u_i$ .

## Our intra-problem $f_u$

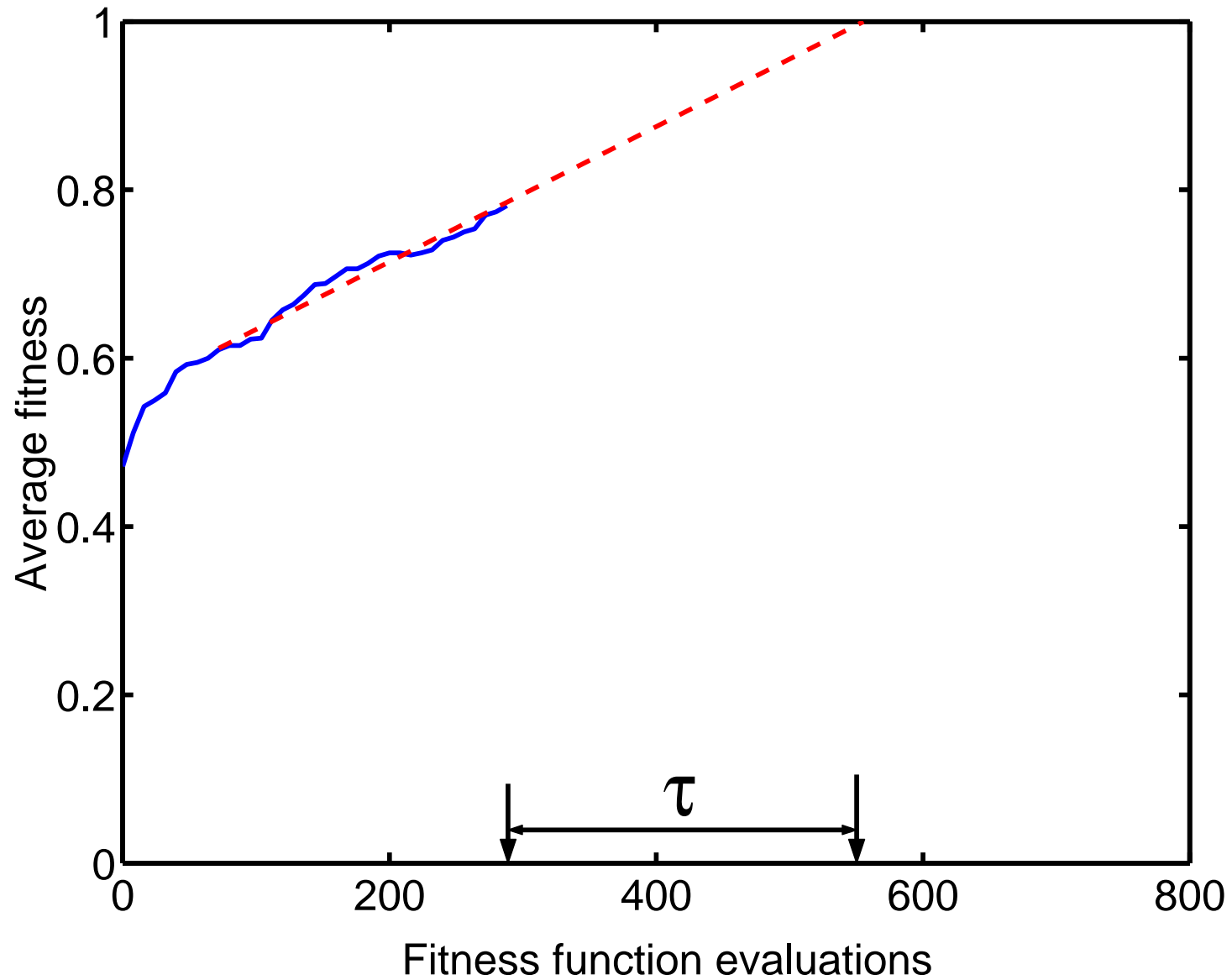
We consider a scalar  $d$ , that has to reach a target value ( $H_i$  is a *learning curve*)

- from  $H_i$ , predict time  $T_{i,sol}$  at which  $d_i$  would reach the target value
- evaluate predicted residual time  $\tau_i = T_{i,sol} - T_i$
- set  $u_i = 1/\tau_i$
- prediction based on a shifting window linear regression
- prediction gets updated at each step
- *intra-problem* approach - learning from a single problem

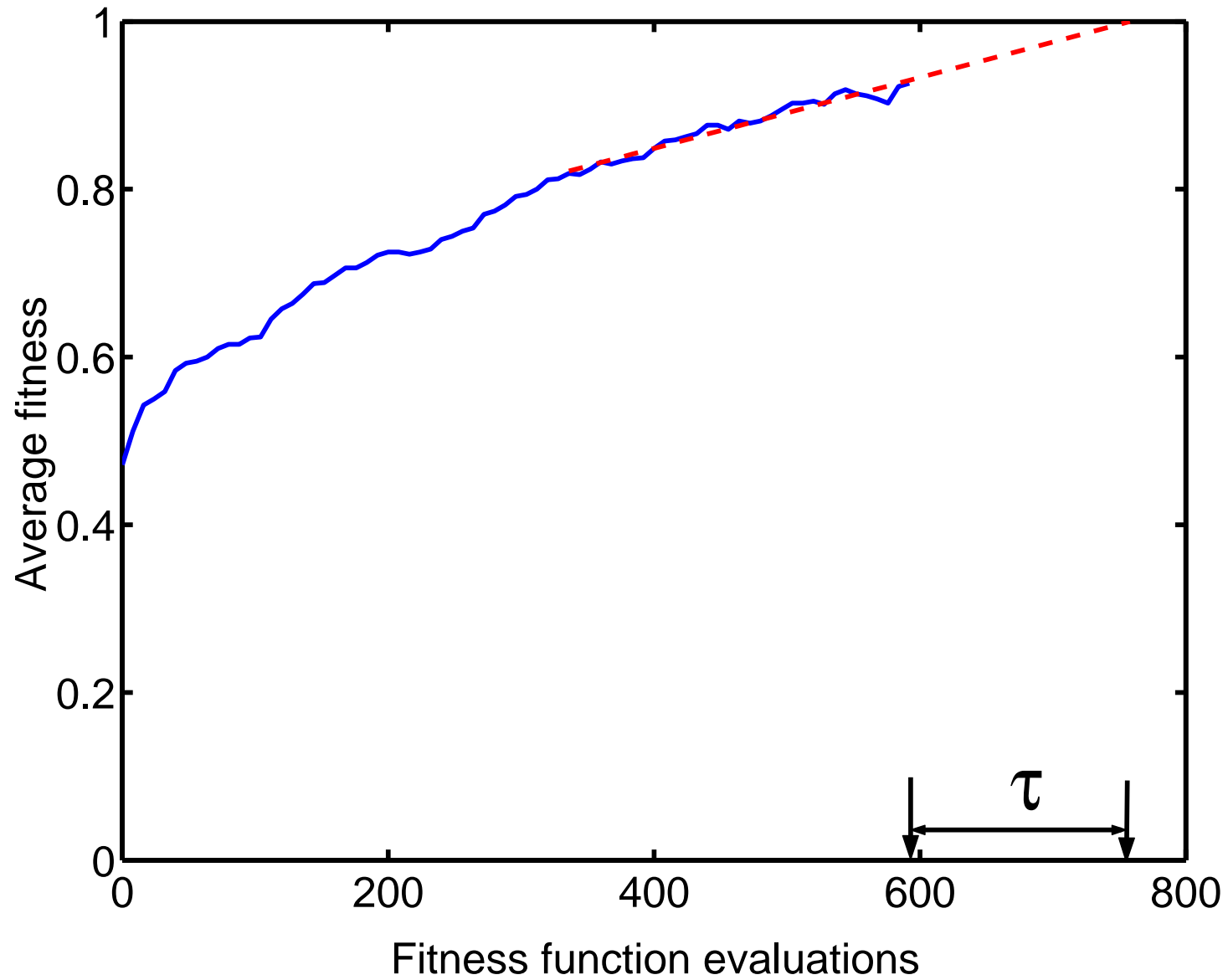
$f_u$  - learning curve and its regression - I



$f_u$  - learning curve and its regression - II



$f_u$  - learning curve and its regression - III



## $f_u$ - details

- noise sensitivity/reaction speed trade-off in choosing size  $c$  of the shifting window used for the regression
- optimal  $c$  may vary across  $A$

Solution: adaptive window sizing

- initialize  $c_i = 2 \forall i$
- keep three models  $\forall i$ , with window sizes  $c_i, 2 * c_i, \max(2, c_i/2)$ .
- at each step, best predictor becomes current  $c_i$

## Choices for $f_P$

- plain normalization of  $u$  values
- greedy approach:  $\Delta T/2$  to current best  $a$ ,  $\Delta T/2$  according to normalized  $u$  values

# Experiments

## Algorithm sets

- $A_1$ : 19 simple generational GAs, no mutation, uniform crossover, population sizes  $2^i, i = 1..19$
- $A_2$ : as  $A_1$  with mutation probabilities 0 and  $0.7/L$  ( $19 \times 2 = 38$  elements)
- $A_3$ : as  $A_2$  with choice between uniform and one-point crossover ( $19 \times 2 \times 2 = 76$  elements)

## Problems

- three CNF3 SAT problems (from SATLIB)
- two synthetic problems from (Harick & Lobo, GECCO '99)
- two TSP instances from TSPLIB

### Comparison term

Parameter-less GA (Harick & Lobo, GECCO '99): an example of algorithm-specific intra-problem AOTA (only applicable to set  $A_1$ )

- $A$ : generational GAs  $a_i$ , population size  $2^i$ , *no* mutation
- each  $a_i$  gets time proportional to  $2^{-i}$ .
- once  $a_i$  converges, or some  $a_j$  ( $j > i$ ) achieves higher average fitness,  $a_i$  is stopped
- time allocation is adjusted such that  $t_i = 2t_{i+1}$  is kept

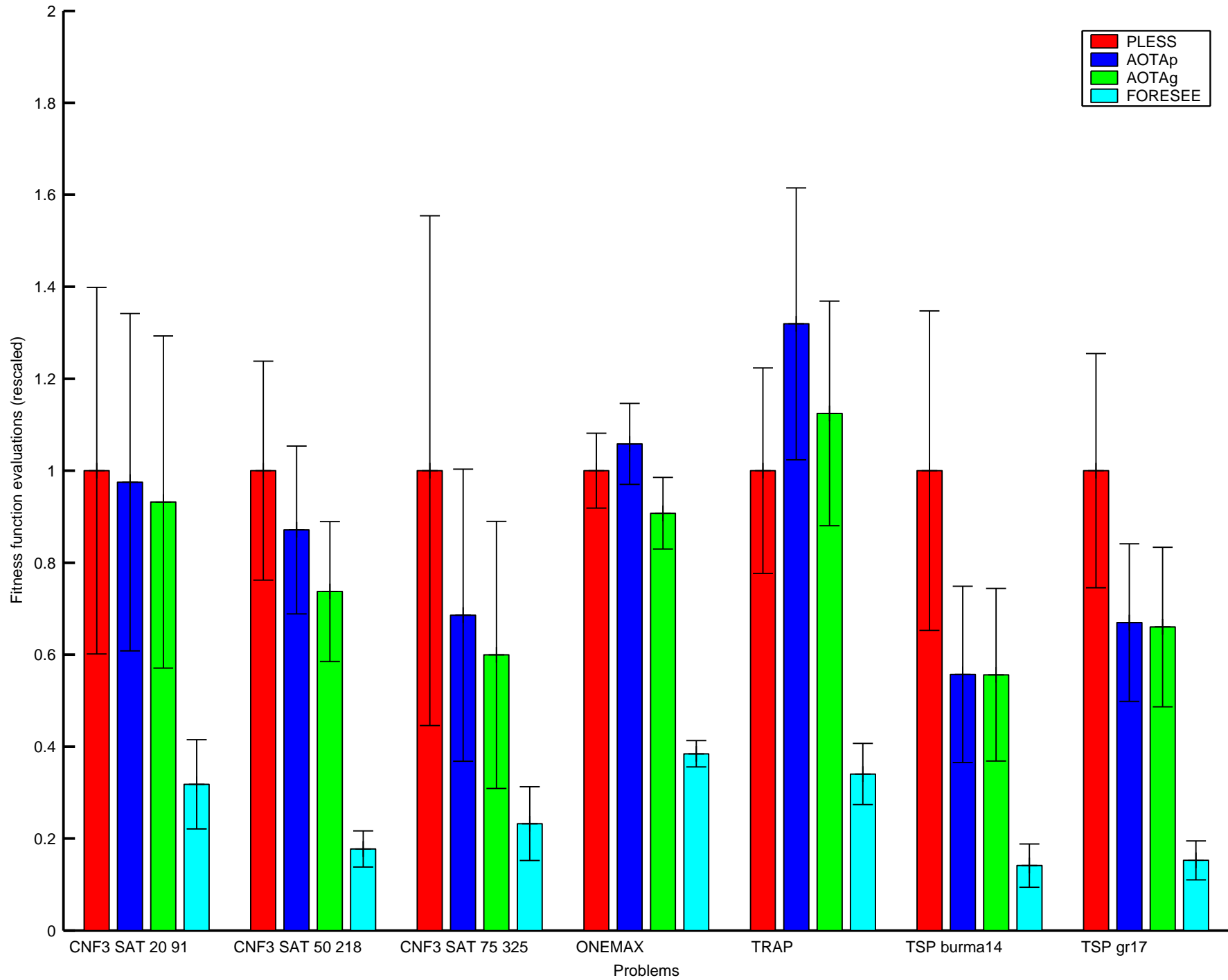
### Results

- $A_1$ : performance comparable to Parameter-less GA (with much less a-priori knowledge)!!
- $A_2, A_3$ : performance gets better when faster algorithms are added, but can worsen if similarly performing algorithms are added. Overhead respect to the fastest element of the set grows almost linearly with  $N$  (see article table). . .
- greedy approach slightly better

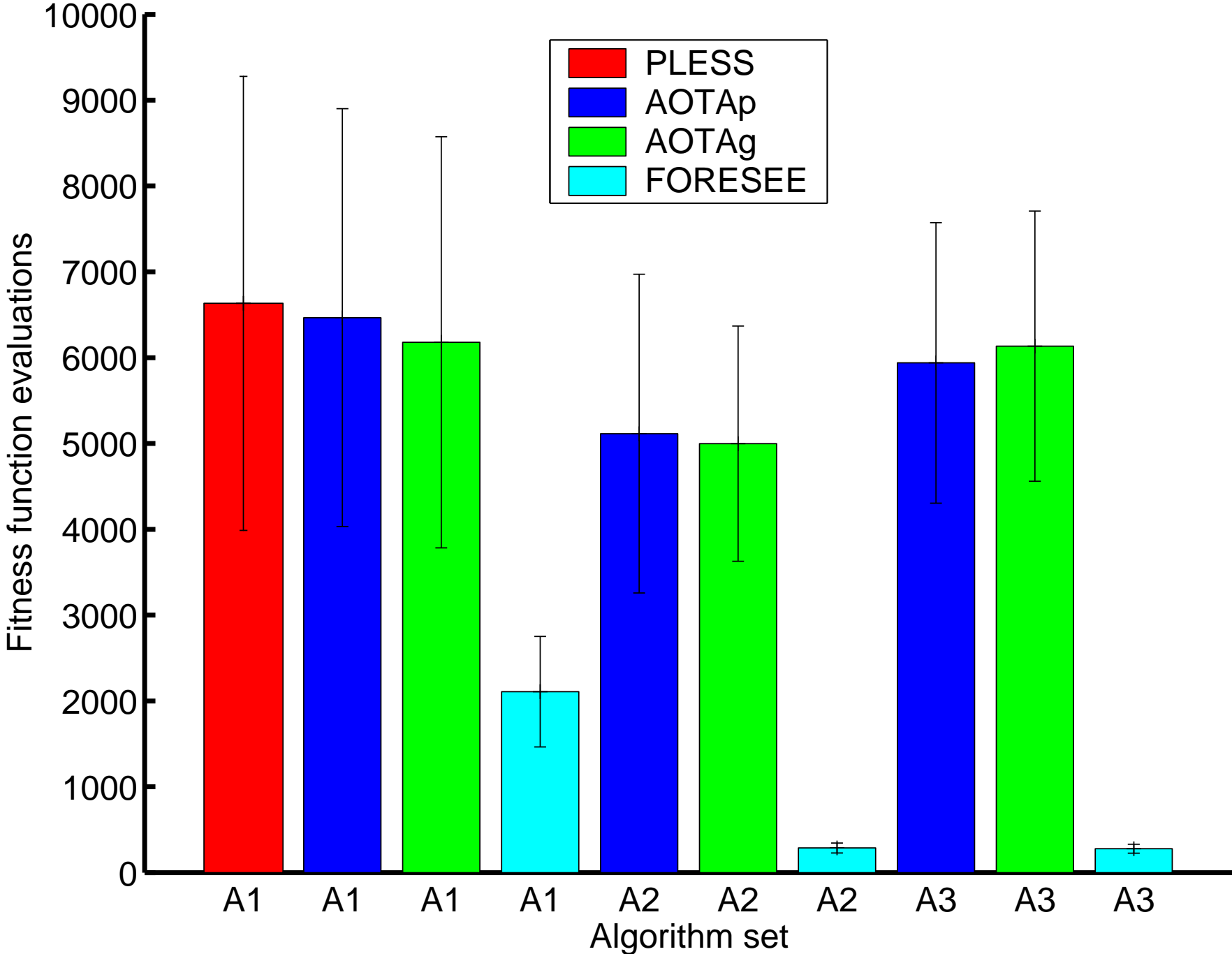
### Legend

- **PLESS**: the Parameterless GA
- **AOTAp**: our AOTA with plain normalization  $f_P$
- **AOTAg**: our AOTA with “greedy”  $f_P$
- **FORESEE**: lower bound: performance of an ideal AOTA with “foresight”, that would be obtained by executing only the (a priori unknown, and usually different for each random seed) fastest element of the algorithm set  $A$ .

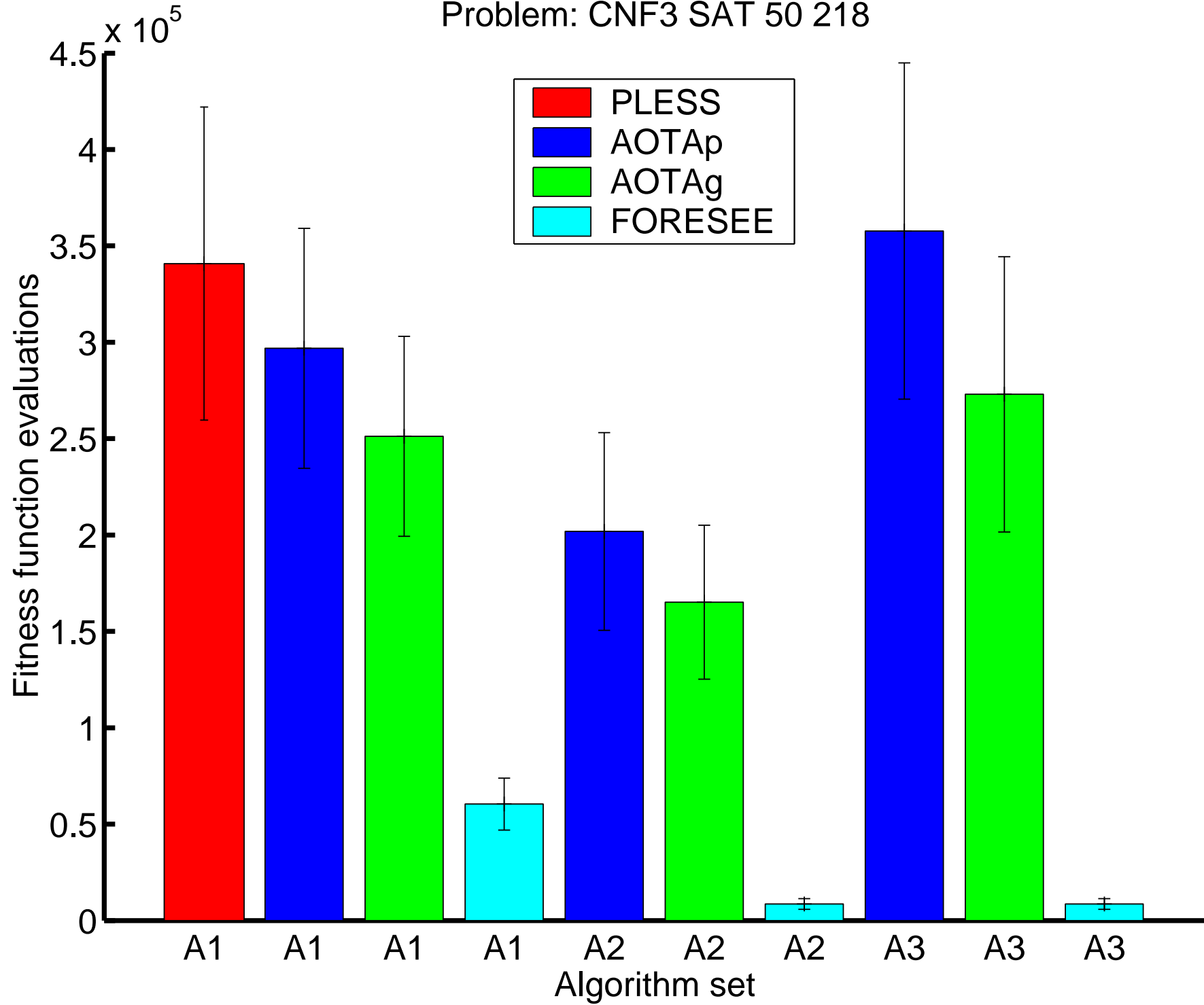
PLESS and our AOTA (plain and greedy  $f_p$ ) on set  $A_1$



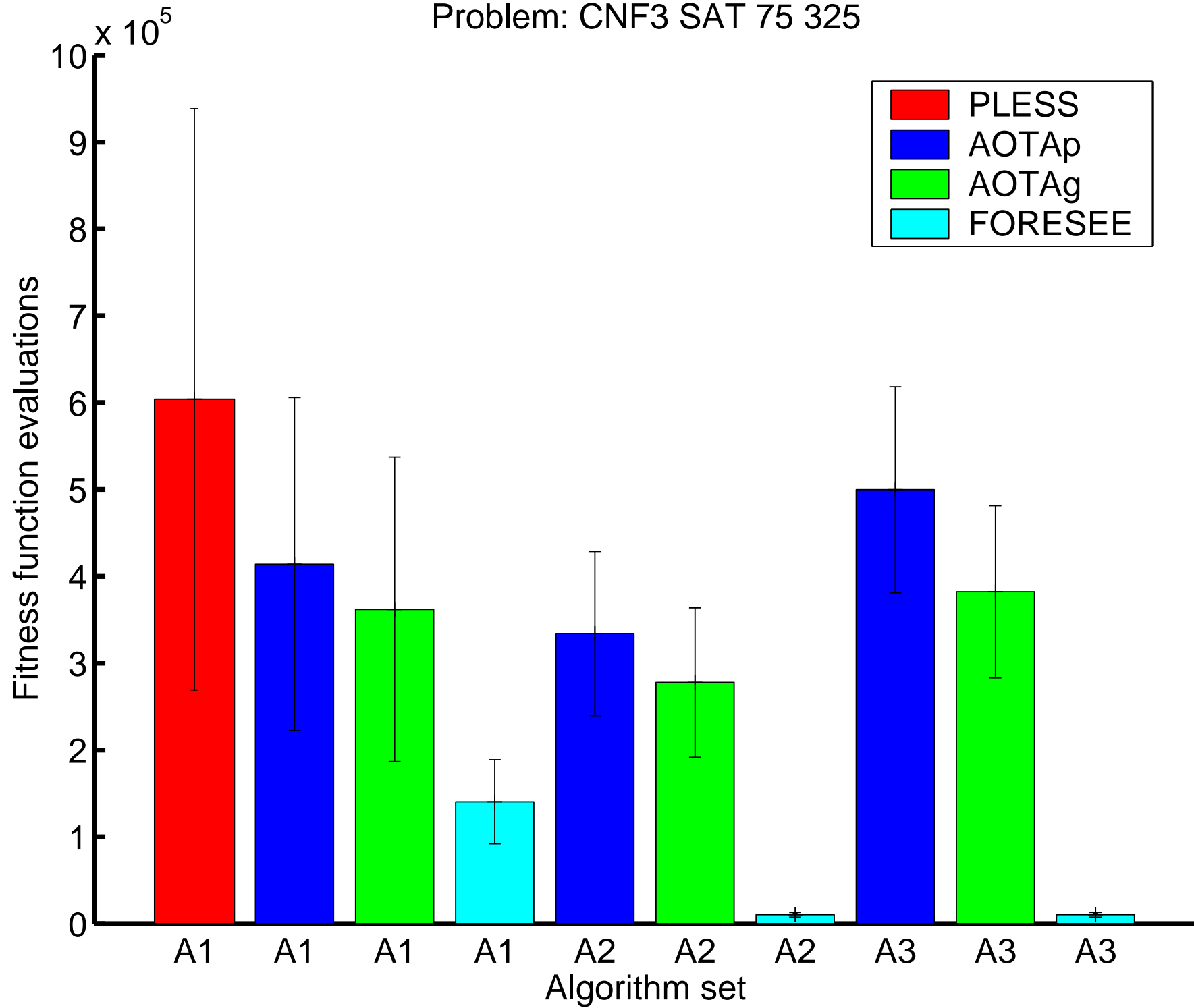
Problem: CNF3 SAT 20 91



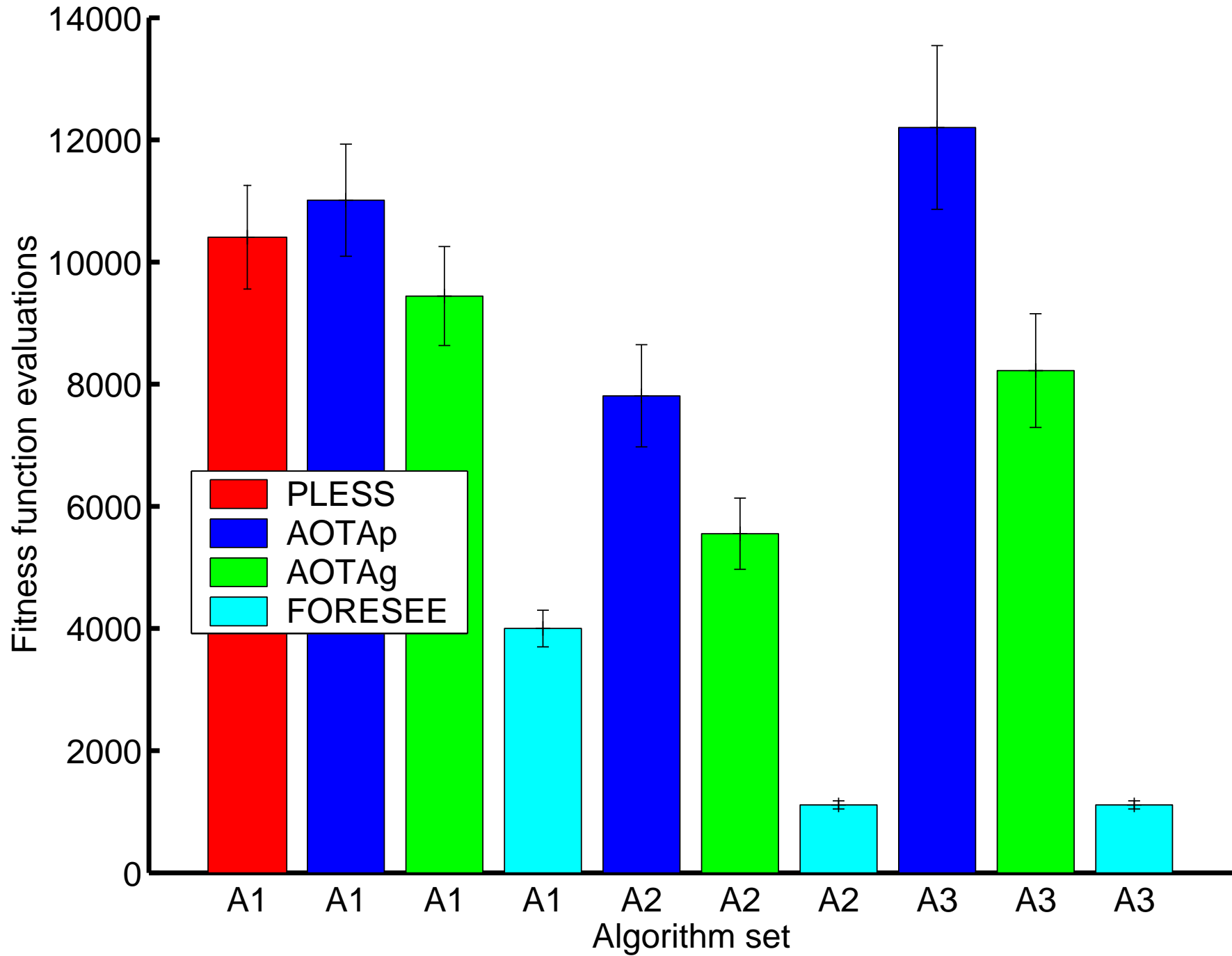
Problem: CNF3 SAT 50 218



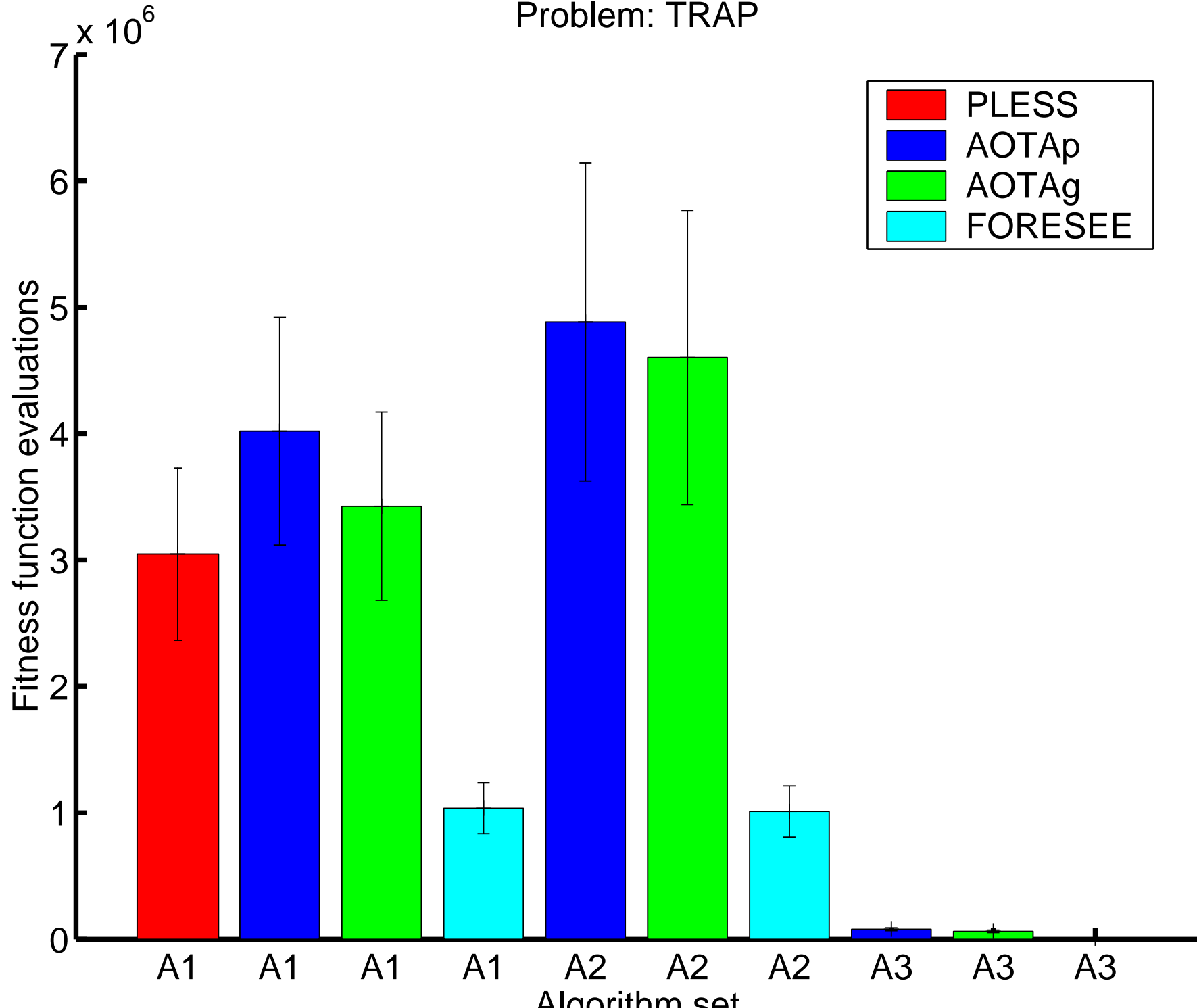
Problem: CNF3 SAT 75 325



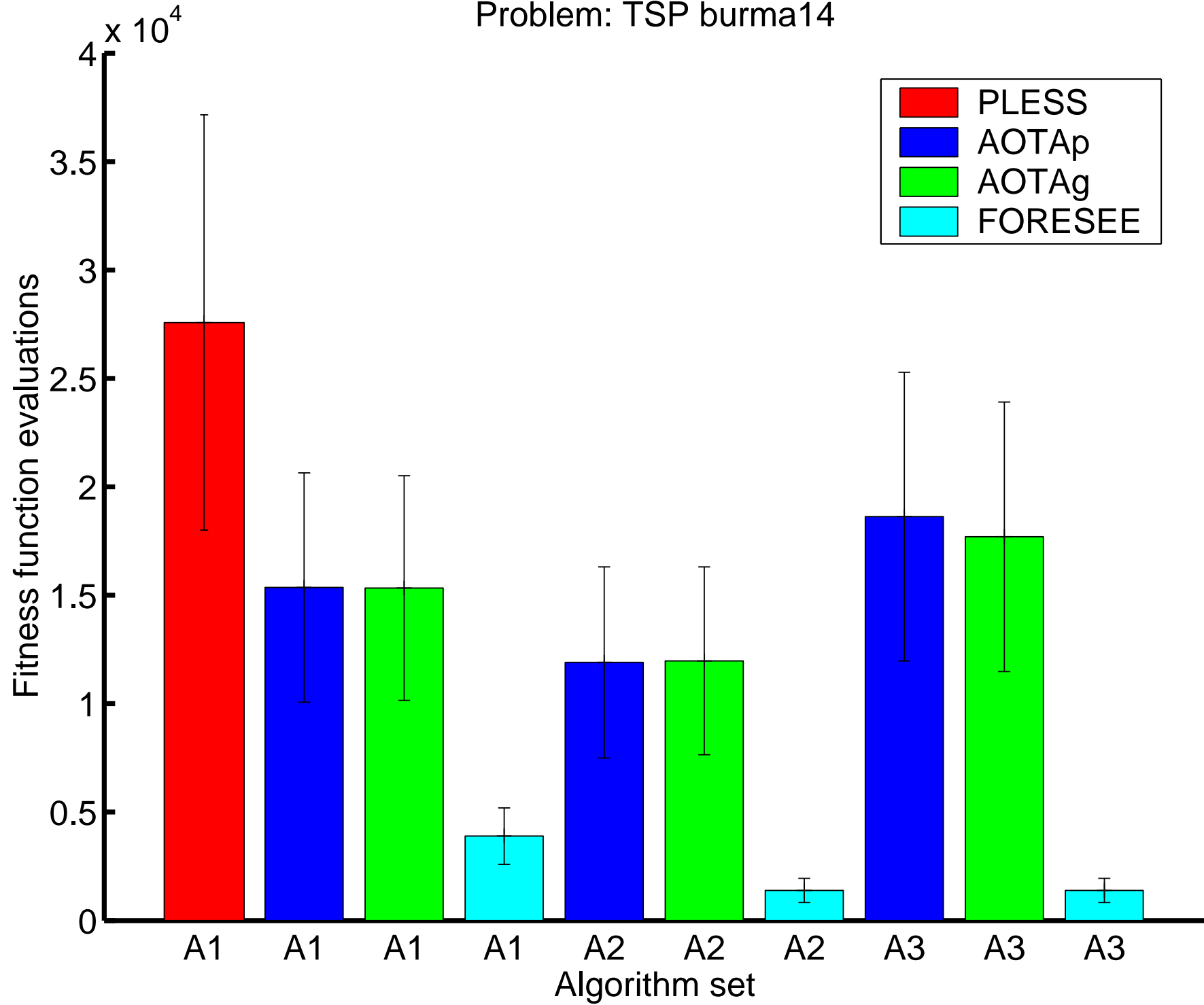
Problem: ONEMAX



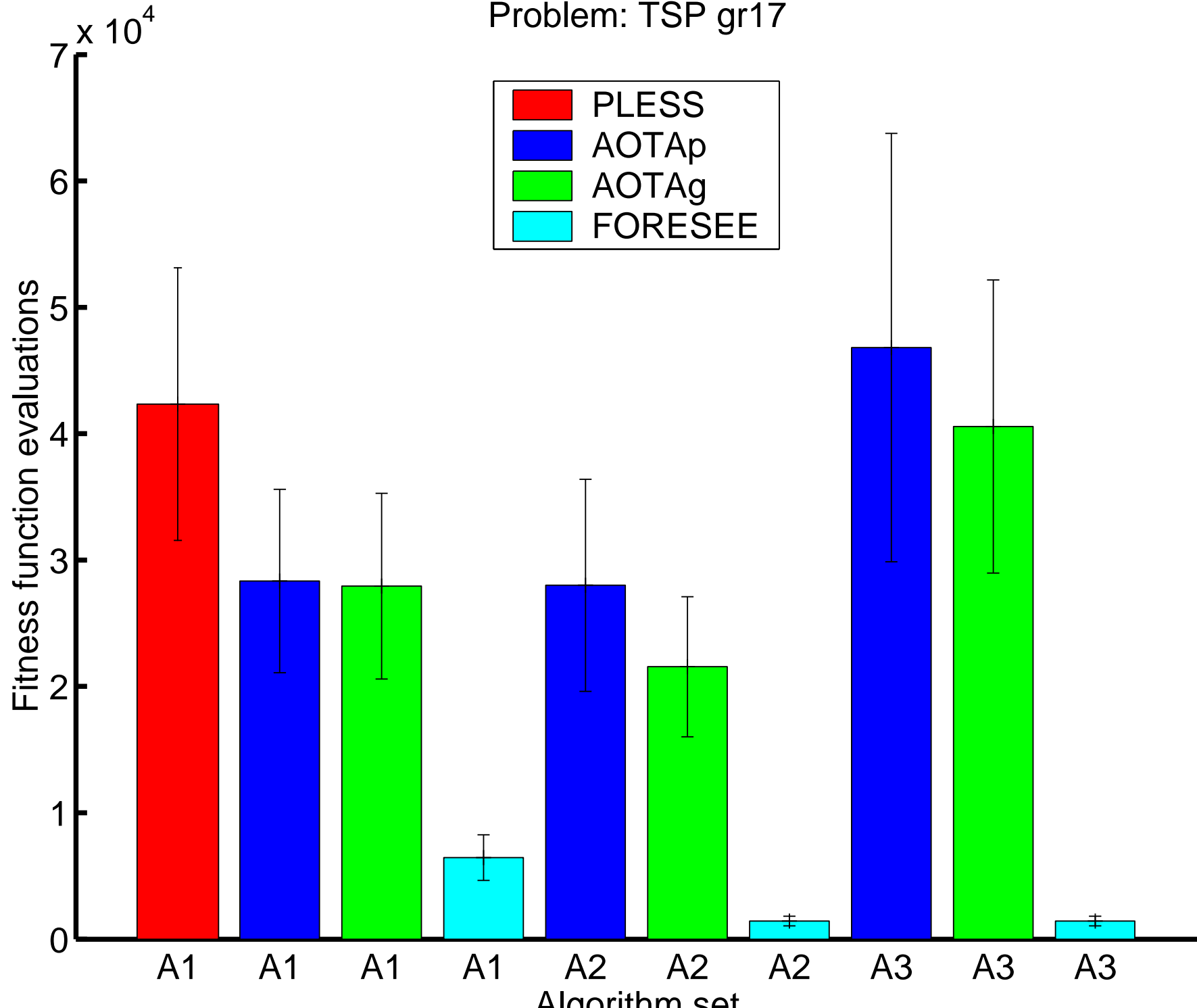
Problem: TRAP



Problem: TSP burma14



Problem: TSP gr17



# Conclusions

## Pros

- a simple, generic intra-problem AOTA gave unexpectedly good results, on a varied set of problems
- the same approach can in principle be extended to any GA parameter
- performance depends solely on the shape of learning curves, so it can in principle be used with other algorithms/problems with similar dynamics

## Cons

- overhead scales badly, esp with similarly performing algorithms
- performance can be bad with “stepwise” learning curves
- choice of  $A$  is left to user

# Future work

Adaptive Online Time Allocation is feasible. We expect a much better performance from *inter-problem* approaches.

Future work will include:

- other choices for  $f_u$
- other choices for  $f_P$ , blocking techniques
- *inter-problem* adaptive  $f_u$ 's
- adaptive  $A$
- application to other algorithms