Metaheuristic Design Problem

Elizabeth Montero Ureta

Computer Science Department
Universidad Técnica Federico Santa María

May 25, 2015

Metaheuristic Design Problem

Motivation

"We are currently involved in a project whose goal is to propose strategies to assist the decision-making process of designers of metaheuristic algorithms. "

Metaheuristic Design Problem

- Incremental development
- Iterative addition of components
- Unknown and complex interactions between components
- Simpler design with equivalent performance ⇒ Efficient metaheuristics

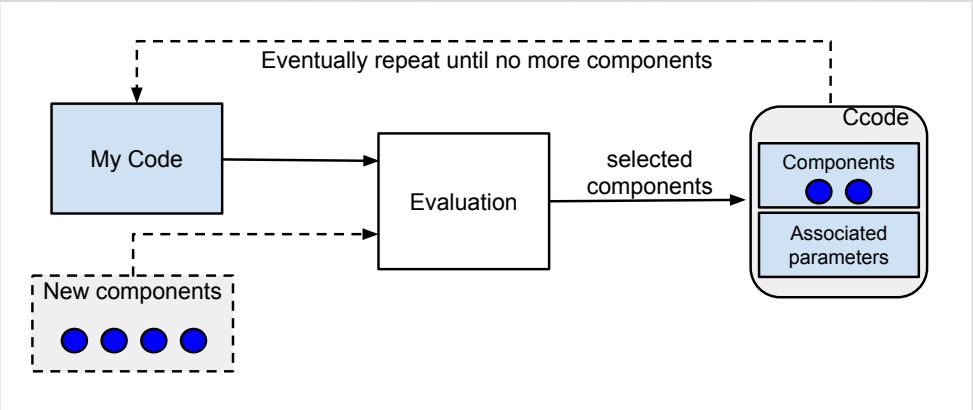


Figure: Design Problems

Metaheuristic Design Problem

- Incremental development
- Iterative addition of components
- Unknown and complex interactions between components
- Simpler design with equivalent performance ⇒ Efficient metaheuristics

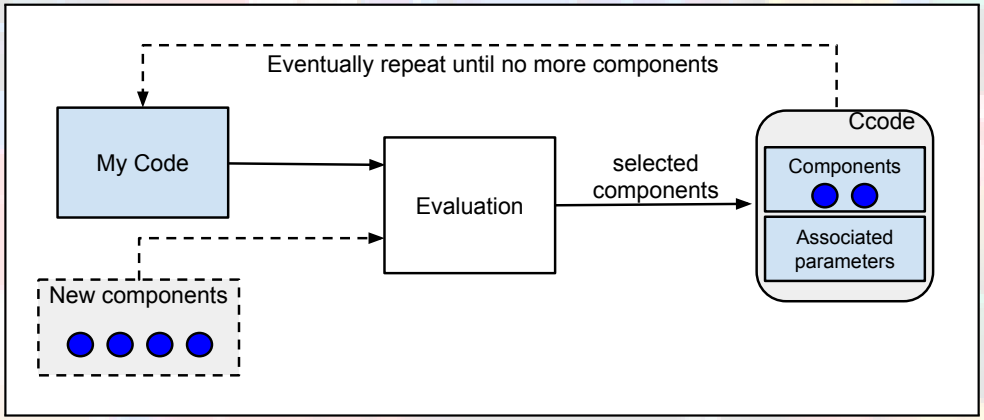


Figure: Design Problems

Stepwise procedures in Multiple Regression

Forward Selection

It starts with an empty model and at each step adds the more informative variable in the presence of the predictors already in the equation.
In *Stepwise regression* variables are removed from the model if they become non significant as other predictors are added.

Backward Elimination

It starts with all of the predictors in the model. The variable that is least significant is removed and the model is refitted. Each subsequent step removes the least significant variable in the model until all remaining variables have demonstrated being enough informative.

Definitions

- On-the-fly Metaheuristic Design problem (OMD)
- Post (Refining) Metaheuristic Design problem (RMD)

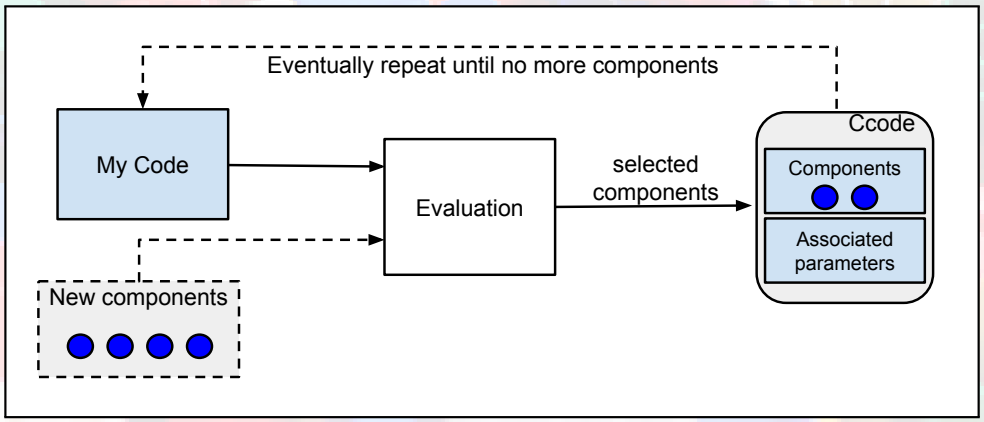


Figure: Design Problems

Definitions (1/2)

On-the-fly Metaheuristic Design problem (OMD)

Given:

- an intermediate design step,
- current code of a metaheuristic $\mathcal{M}$ and
- a set of candidate components $S = \{C_1, \ldots, C_n\}$ to be included in $\mathcal{M}$

The OMD consists in finding a code $\mathcal{M}^*$ using the selected components among the already included and the new candidates, in order to improve the performance of $\mathcal{M}$.

Definitions (1/2)

On-the-fly Metaheuristic Design problem (OMD)

Given:

- an intermediate design step,
- current code of a metaheuristic $\mathcal{M}$ and
- a set of candidate components $S = \{C_1, \ldots, C_n\}$ to be included in $\mathcal{M}$

The OMD consists in finding a code $\mathcal{M}^*$ using the selected components among the already included and the new candidates, in order to improve the performance of $\mathcal{M}$.

Definitions (2/2)

Post (Refining) Metaheuristic Design problem (RMD)

Given:

- a target metaheuristic $\mathcal{M}$,
- a set of parameters for $\mathcal{M}$ and
- a set of input data

The RMD consists in finding a reduced code $\mathcal{M}^{\mathcal{R}}$ which gives, at least, the same performance than $\mathcal{M}$ with the same input data.

Definitions (2/2)

Post (Refining) Metaheuristic Design problem (RMD)

Given:

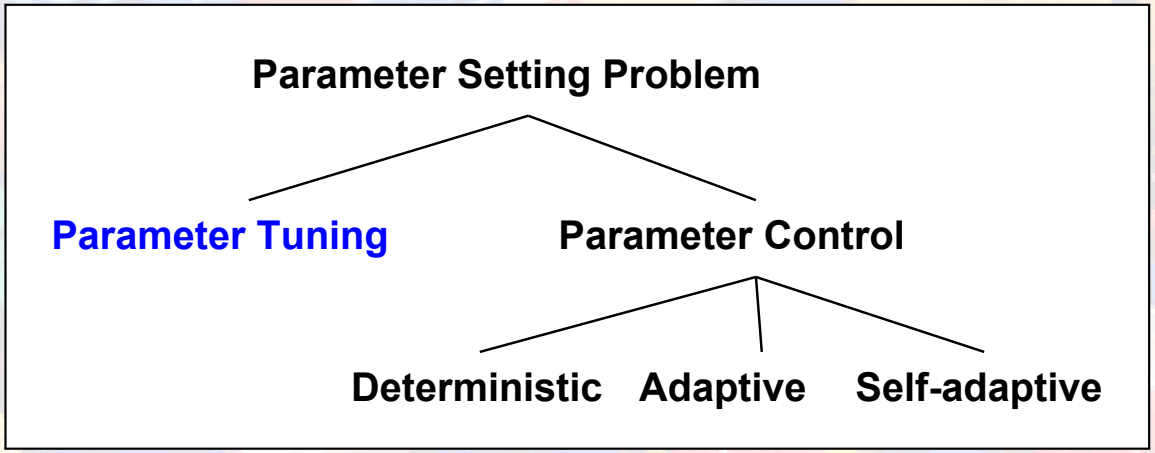
- a target metaheuristic $\mathcal{M}$,
- a set of parameters for $\mathcal{M}$ and
- a set of input data

The RMD consists in finding a reduced code $\mathcal{M}^{\mathcal{R}}$ which gives, at least, the same performance than $\mathcal{M}$ with the same input data.

Strategy to use tuners for designing metaheuristics

Tuning methods

- Taxonomy



- Finding a (a set of) parameter configuration
- Features
  - Before the execution of the target metaheuristic.
  - Execute target metaheuristic several times.
  - Time consuming.
  - Keep parameter values fixed during the target metaheuristic execution.

- Numerical parameters: Large number of possible values
- Categorical parameters: Discrete number of possible values, no distance metric

Automated tuning process

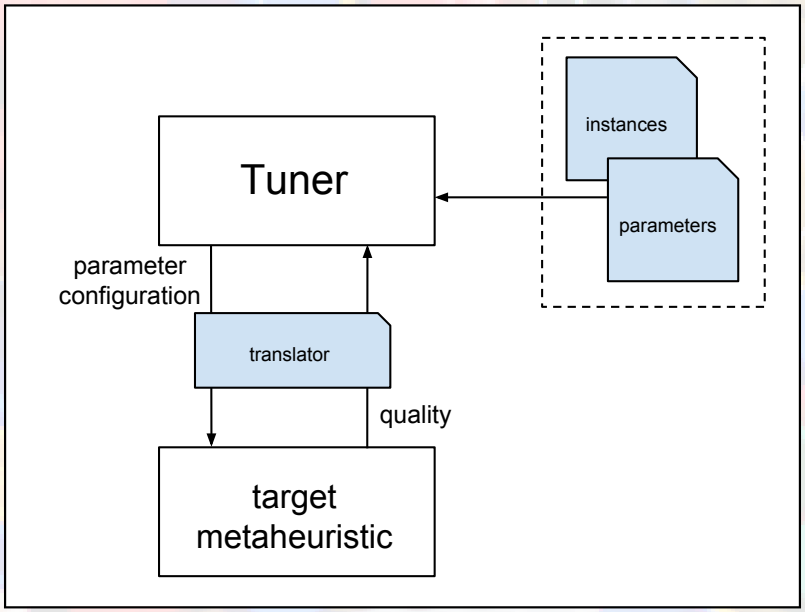


Figure: Automated tuning process

Collective Strategy

- Generate a competition between the components being evaluated.
- Use the tuner to support the decision process.

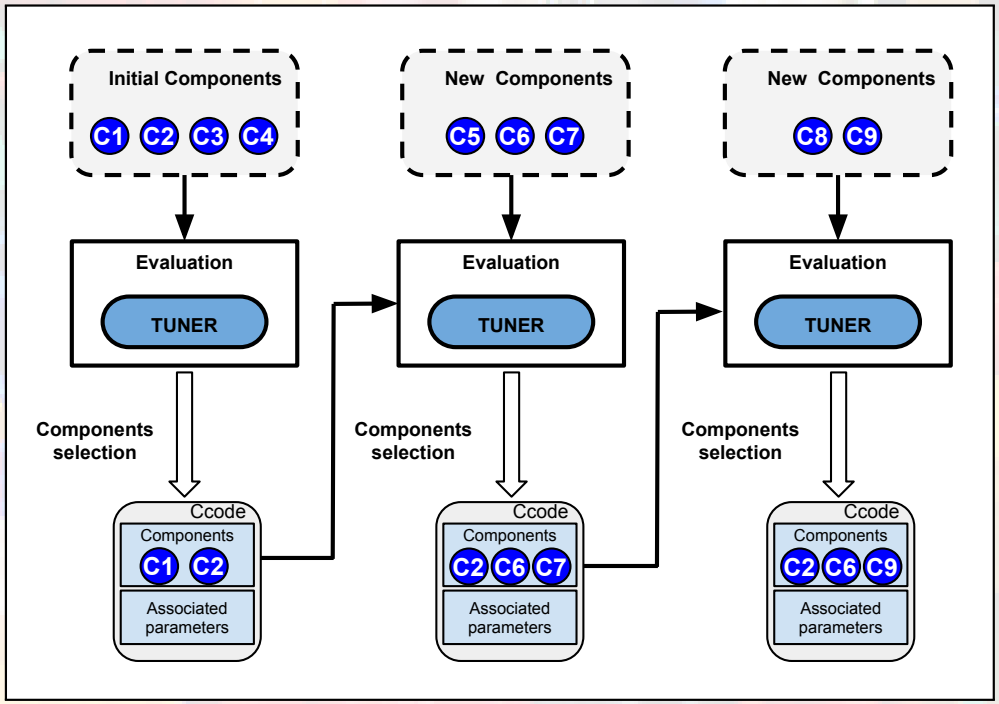


Figure: Collective Strategy

Experiments

Evaluation of Different Design Options

Detection of ineffective operators [Montero et al., 2012]

- Goal: Evaluate F-RACE, Paramils and REVAC for detecting ineffective operators
- Operator rate of dummy operator in Ant Solver for CSP's [Solnon, 2002]

Selection of best components [Montero and Riff, 2014]

- Goal: Proposing EVOCA and compare its results with I-RACE
- Selection of operators in GA for NK landscapes [Pelikan, 2008].
- Design of mutation operator in AIS for MO [Pierrard and Coello Coello, 2012]

Determination of best (hyper-) mutation operators [Montero and Riff, 2015]

- Goal: Using EVOCA to detect important components in more sophisticated algorithms
- Selection of mutation operators in AIS for RTTP [Pérez-Cáceres and Riff, 2015].

Tuning Methods bibliography

- F-Race [Birattari et al., 2002] and I-Race [Birattari et al., 2010]
- REVAC (Relevance Estimation and VAlue Calibration) [Nannen and Eiben, 2007]
- ParamILS (Parameter Iterated Local Search) [Hutter et al., 2007]
- SPO (Sequential Parameter Optimization) [Bartz-Beielstein et al., 2005]
- EVOCA (EVolutionary CAlibrator) [Riff and Montero, 2013]

EVOCA [Riff and Montero, 2013]

Extend the capabilities of tuners from tuning parameter values to supporting the design process of metaheuristics.

Main features

- It does not require the discretization of parameter ranges of values
  - able to search the entire set of possible values
  - able to increase the precision of real valued parameters.
- It focuses in determining alternative quality parameter configurations.
  - supporting the metaheuristic design process.

EVOCA [Riff and Montero, 2013]

Extend the capabilities of tuners from tuning parameter values to supporting the design process of metaheuristics.

Main features

- It does not require the discretization of parameter ranges of values
  - able to search the entire set of possible values
  - able to increase the precision of real valued parameters.
- It focuses in determining alternative quality parameter configurations.
  - supporting the metaheuristic design process.

EVOCA [Riff and Montero, 2013]

Description

- Steady state Evolutionary Algorithm
  - Population of parameter configurations
- Initial population [LHD]
  - Min, Max and a set of intermediate values in [Min, Max]
- Quality function
  - Each configuration is executed NumSeeds times
- At most two new configurations per iteration
  - Wheel-based multi-parent crossover
  - Hill-climbing first-improvement rule.

EVOCA tuner

Algorithm 1: EVOCA tuner

Input : Target algorithm $\mathcal{A}$ and set of parameters to tune
Output: Best performing parameter configurations for $\mathcal{A}$

```
1  GenerateInitialPopulation (P);
2  Evaluate(P);
3  while Stopping criterion is not met do
4  |   child ← Wheel-cross(P);
5  |   Evaluate(child,R);
   |   /*Replace the worst configuration in P by the new one        */
6  |   ReplaceWorst(P,child);
7  |   mutated_child ← Mutate(child);
8  |   Evaluate(mutated_child,R);
9  |   if mutated_child is better than child then
   |   |   /*Replace the second worst configuration in P only if
   |   |     mutated child is better than child                     */
10 |   |   ReplaceSecondWorst(P,mutated_child);
11 |   end
12 end
13 return P;
```

Traveling Tournament Problem (TTP)

Traveling Tournament Problem

- Double round robin tournament: n teams need $2 * (n - 2)$ slots
- No more than three consecutive home or three consecutive road games
- No repeaters (A at B, followed immediately by B at A)
- Minimize distance traveled
  - Teams begin in their home city and must return there after the tournament

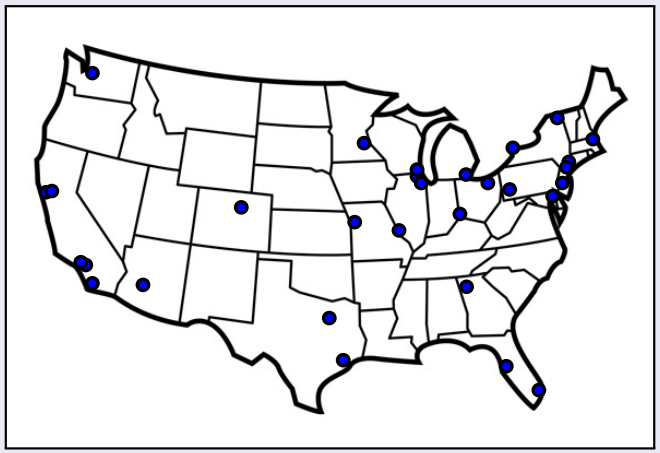


Figure: Baseball National League

Traveling Tournament Problem (TTP)

Traveling Tournament Problem

- Double round robin tournament: n teams need $2 * (n - 2)$ slots
- No more than three consecutive home or three consecutive road games
- No repeaters (A at B, followed immediately by B at A)
- Minimize distance traveled
  - Teams begin in their home city and must return there after the tournament

Relaxed TTP

- Teams are allowed to have a fixed number K of byes, hence RTTP schedule length is $2 * (n - 1) + K$.
- Byes are ignored in determining the length of a home stand or road trip, and in determining whether a repeater has occurred.
- The higher the value of K the more flexibility to find schedules with lower distance traveled.

Scenario [Montero and Riff, 2015]

Artificial Immune System for RTTP

- Target metaheuristic
  - AIS works with a population of matrices that represent tournaments
  - New solutions are created by applying a set of (hyper-) mutation operators: (1) SwapTeams, (2) SwapHomes, (3) SwapRounds, (4) PartialSwapTeams, (5) PartialSwapRounds, (6) SwapByesMatches and (7) GroupingAwayByes.
  - Minimization of total distance traveled.
- Problem instances
  - Baseball National League instances ranging from 4 to 16 teams
  - Considering byes ranging from $K = 1$ to $K = 3$
- Tuning process
  - Budget of 10000 executions of target algorithm
  - 20 tuner executions
  - Parameters
    - 7 binary parameters for (hyper-) mutation operators
    - cl_{rate}: percentage of solutions to be selected for cloning,
    - cl_{factor}: clonal factor related to the amount of clones that will be generated from each solution and
    - r_{rate}: percentage of new solutions included each iteration

Scenario [Montero and Riff, 2015]

Artificial Immune System for RTTP

- Target metaheuristic
  - AIS works with a population of matrices that represent tournaments
  - New solutions are created by applying a set of (hyper-) mutation operators: (1) SwapTeams, (2) SwapHomes, (3) SwapRounds, (4) PartialSwapTeams, (5) PartialSwapRounds, (6) SwapByesMatches and (7) GroupingAwayByes.
  - Minimization of total distance traveled.
- Problem instances
  - Baseball National League instances ranging from 4 to 16 teams
  - Considering byes ranging from $K = 1$ to $K = 3$
- Tuning process
  - Budget of 10000 executions of target algorithm
  - 20 tuner executions
  - Parameters
    - 7 binary parameters for (hyper-) mutation operators
    - cl_{rate}: percentage of solutions to be selected for cloning,
    - cl_{factor}: clonal factor related to the amount of clones that will be generated from each solution and
    - r_{rate}: percentage of new solutions included each iteration

Scenario [Montero and Riff, 2015]

Artificial Immune System for RTTP

- Target metaheuristic
  - AIS works with a population of matrices that represent tournaments
  - New solutions are created by applying a set of (hyper-) mutation operators: (1) SwapTeams, (2) SwapHomes, (3) SwapRounds, (4) PartialSwapTeams, (5) PartialSwapRounds, (6) SwapByesMatches and (7) GroupingAwayByes.
  - Minimization of total distance traveled.
- Problem instances
  - Baseball National League instances ranging from 4 to 16 teams
  - Considering byes ranging from $K = 1$ to $K = 3$
- Tuning process
  - Budget of 10000 executions of target algorithm
  - 20 tuner executions
  - Parameters
    - 7 binary parameters for (hyper-) mutation operators
    - cl_{rate}: percentage of solutions to be selected for cloning,
    - cl_{factor}: clonal factor related to the amount of clones that will be generated from each solution and
    - r_{rate}: percentage of new solutions included each iteration

(hyper-) mutation process

Algorithm 2: (hyper-) mutation process

```
1  forall the s ∈ P_c do
       /*Ranking to determine hypermutation level                    */
2      for x ← 1 to ranking do
3          for y ← 1 to 6 do
4              s*_y ← op_y(s);
5          end
6          sol ← Best(s*_1, ..., s*_6);
7          s*_7 ←GroupingAwayByes(sol);
8          if Better(s*_7, sol) then
9              sol ← s*_7;
10         end
11     end
12     s ← sol;
13 end
14 return;
```

Refined (hyper-) mutation process

Algorithm 3: Refined (hyper-) mutation process

```
1  forall the s ∈ P_c do
       /*Ranking to determine hypermutation level                    */
2      for x ← 1 to ranking do
3          for y ← 1 to 6 do
               /*Value of each binary parameter determines the use of
                 related move                                        */
4              if UseOp_y then
5                  s*_y ← op_y(s);
6              end
7          end
8          sol ← Best(s*_1, ..., s*_6);
           /*Value of parameter UseGAB determines the use of
             GroupingAwayByes                                        */
9          if UseGAB then
10             s*_7 ←GroupingAwayByes(sol);
11         end
12         if Better(s*_7, sol) then
13             sol ← s*_7;
14         end
15     end
16     s ← sol;
17 end
18 return;
```

Results [Montero and Riff, 2015]

Id.	*cl_rate*	*cl_factor*	r_{rate}	*useST*	*useSH*	*useSR*	*usePST*	*usePSR*	*useSBM*	*useGA*	$G_P(\theta)$
E0	0.90	0.80	0.32	1	0	0	1	1	0	1	1.96
E1	0.82	0.30	0.00	1	1	1	1	1	1	1	2.09
E2	1.00	0.20	0.30	1	1	1	1	1	0	1	2.19
E3	0.89	0.18	0.53	1	1	0	1	1	0	1	2.49
E4	0.60	1.00	0.40	1	1	0	1	1	1	1	1.09
E5	0.80	1.00	0.28	1	1	0	1	1	1	0	2.13
E6	1.00	1.00	0.20	1	1	0	0	1	0	1	1.19
E7	0.50	0.90	0.60	1	1	0	1	1	0	1	1.80
E8	0.40	0.90	0.05	1	1	0	1	1	0	1	0.85
E9	0.50	1.00	0.20	1	0	0	1	1	1	1	1.83
E10	0.84	0.90	0.50	1	1	1	1	1	0	0	2.37
E11	0.51	0.97	0.00	1	1	0	0	1	0	1	1.01
E12	**0.78**	**1.00**	**0.20**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0.62**
E13	0.80	0.50	0.00	0	1	0	1	1	1	0	5.34
E14	0.84	0.65	0.56	1	1	0	1	1	0	1	1.77
E15	0.91	0.19	0.20	1	0	1	1	1	1	1	2.97
E16	0.80	0.40	0.50	1	1	0	1	1	1	0	3.52
E17	0.40	0.85	0.00	1	0	1	0	1	1	1	2.96
E18	1.00	0.90	0.00	1	1	1	1	1	0	1	0.87
E19	0.70	0.30	0.60	0	1	1	1	1	0	0	5.60
% usage				90	80	35	85	100	40	75	

Table: Designs and their performances

Results [Montero and Riff, 2015]

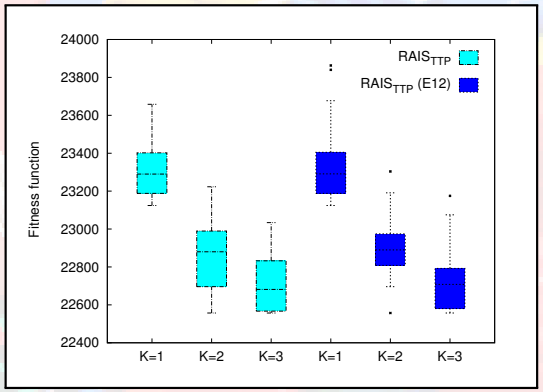
Id.	*cl_rate*	*cl_factor*	r_{rate}	*useST*	*useSH*	*useSR*	*usePST*	*usePSR*	*useSBM*	*useGA*	$G_P(\theta)$
E0	0.90	0.80	0.32	1	0	0	1	1	0	1	1.96
E1	0.82	0.30	0.00	1	1	1	1	1	1	1	2.09
E2	1.00	0.20	0.30	1	1	1	1	1	0	1	2.19
E3	0.89	0.18	0.53	1	1	0	1	1	0	1	2.49
E4	0.60	1.00	0.40	1	1	0	1	1	1	1	1.09
E5	0.80	1.00	0.28	1	1	0	1	1	1	0	2.13
E6	1.00	1.00	0.20	1	1	0	0	1	0	1	1.19
E7	0.50	0.90	0.60	1	1	0	1	1	0	1	1.80
E8	0.40	0.90	0.05	1	1	0	1	1	0	1	0.85
E9	0.50	1.00	0.20	1	0	0	1	1	1	1	1.83
E10	0.84	0.90	0.50	1	1	1	1	1	0	0	2.37
E11	0.51	0.97	0.00	1	1	0	0	1	0	1	1.01
E12	**0.78**	**1.00**	**0.20**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0.62**
E13	0.80	0.50	0.00	0	1	0	1	1	1	0	5.34
E14	0.84	0.65	0.56	1	1	0	1	1	0	1	1.77
E15	0.91	0.19	0.20	1	0	1	1	1	1	1	2.97
E16	0.80	0.40	0.50	1	1	0	1	1	1	0	3.52
E17	0.40	0.85	0.00	1	0	1	0	1	1	1	2.96
E18	1.00	0.90	0.00	1	1	1	1	1	0	1	0.87
E19	0.70	0.30	0.60	0	1	1	1	1	0	0	5.60
% usage				90	80	35	85	100	40	75	

Table: Designs and their performances

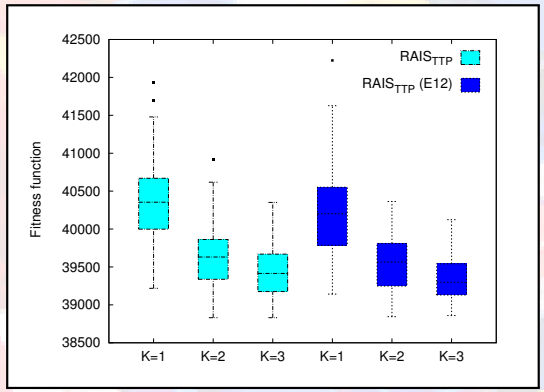
Results [Montero and Riff, 2015]

Instance		$RAIS_{TTP}$		$RAIS_{TTP}(E12)$		p-value
		Average	Best	Average	Best	
NL4	K=1	8160.0	8160	8160.0	8160	-
	K=2	8160.0	8160	8160.0	8160	-
	K=3	8044.0	8044	8044.0	8044	-
NL6	K=1	23308.7	23124	23322.9	23124	0.75
	K=2	22861.9	22557	22895.9	22557	0.08
	K=3	22700.5	22557	22705.7	22557	0.89
NL8	K=1	40343.7	39218	40213.9	39142	0.09
	K=2	39625.1	38831	39569.9	38845	0.29
	K=3	39432.0	38831	39355.7	38859	0.08
NL10	K=1	63542.8	59686	**63160.8**	58825	**0.01**
	K=2	62653.7	60584	**62095.6**	60477	**0.00**
	K=3	61526.9	59486	61407.5	58834	0.48
NL12	K=1	124457.4	120016	124107.1	119570	0.08
	K=2	123027.6	118674	**122089.0**	118037	**0.00**
	K=3	120564.7	115755	**119955.7**	116942	**0.00**
NL14	K=1	224272.7	216236	**222197.0**	208823	**0.00**
	K=2	219010.7	209402	218020.0	209331	0.09
	K=3	215872.1	207495	**213782.1**	206134	**0.00**
NL16	K=1	312448.2	301232	**309895.2**	300531	**0.00**
	K=2	304606.7	297936	**301840.7**	293575	**0.00**
	K=3	299642.1	290998	**296065.9**	284982	**0.00**

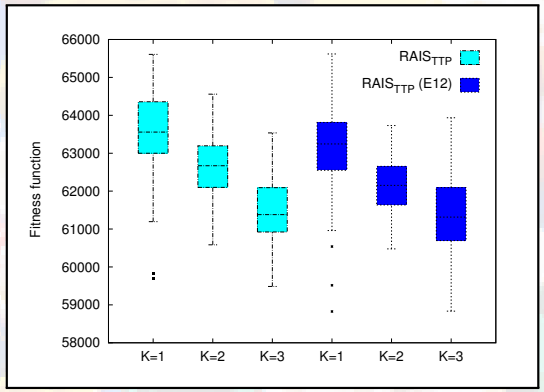
Results [Montero and Riff, 2015]



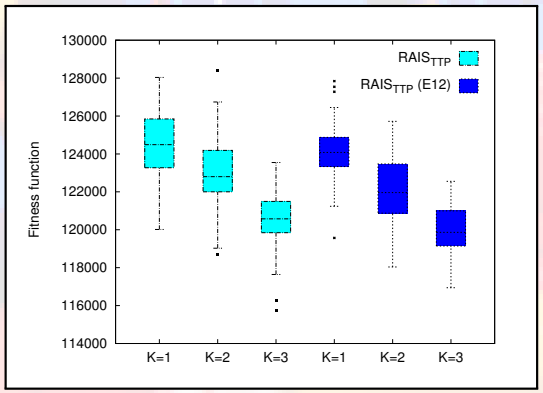
(a) NL6



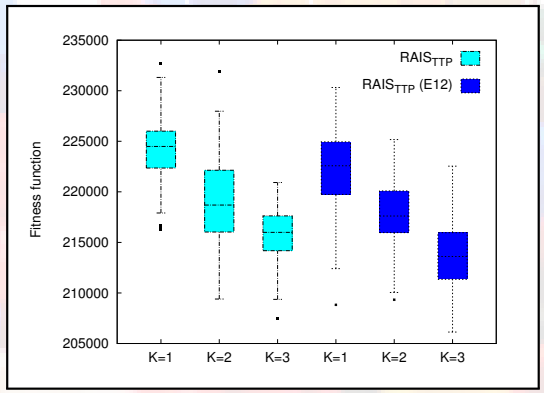
(b) NL8



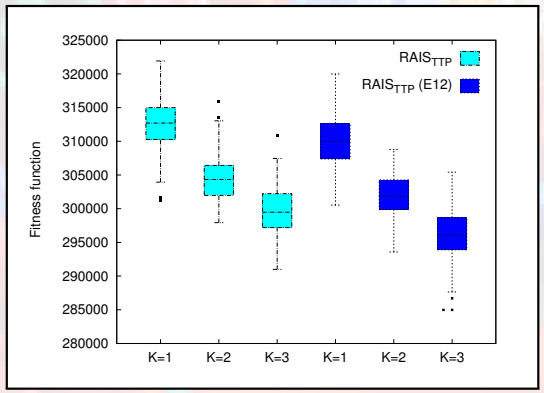
(c) NL10



(d) NL12



(e) NL14

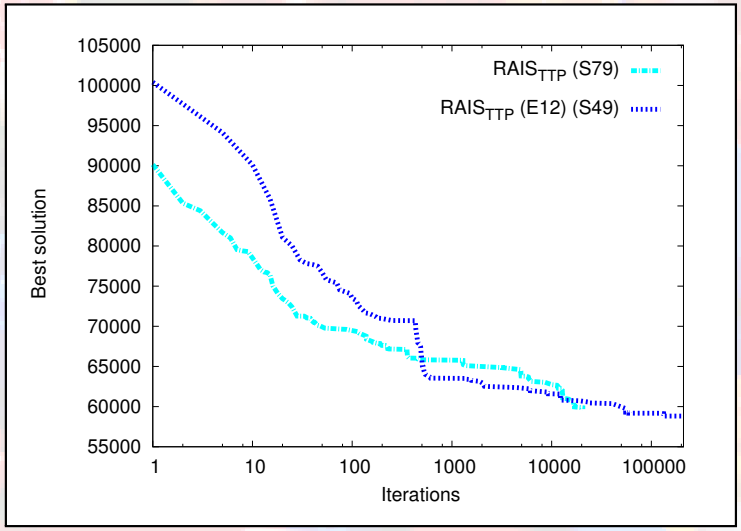


(f) NL16

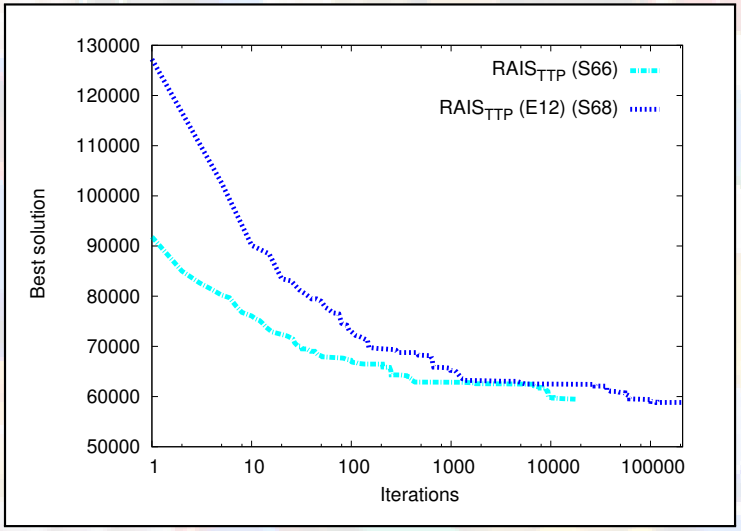
Results [Montero and Riff, 2015]

Instance		Best known	E12	
			Average	Best
NL4	K=1	8160	8160.0	8160
	K=2	8160	8160.0	8160
	K=3	8044	8044.0	8044
NL6	K=1	22642	23322.9	23124
	K=2	22557	22895.9	22557
	K=3	22557	22705.7	22557
NL8	K=1	39128	40213.9	39142
	K=2	38761	39569.9	38845
	K=3	38670	39355.7	38859
NL10	K=1	59425	63160.8	**58825**
	K=2	59373	62095.6	60477
	K=3	59436	61407.5	**58834**
NL12	K=1	108629	124107.1	119570
	K=2	108629	122089.0	118037
	K=3	108629	119955.7	116942
NL14	K=1	183354	222197.0	208823
	K=2	183354	218020.0	209331
	K=3	183354	213782.1	206134
NL16	K=1	249477	309895.2	300531
	K=2	249477	301840.7	293575
	K=3	249477	296065.9	284982

Results [Montero and Riff, 2015]



(a) K=1



(b) K=3

Conclusions

Conclusions [Montero and Riff, 2015]

- In-depth analysis of the immune algorithm $RAIS_{TTP}$
- Simpler code: $RAIS_{TTP}$ ($E12$) that shows no statistical significance difference in performance respect to the original.
- New bounds for the RTTP using the refined code
  - NL10 with K=1 and K=3.

- Development of tools for facing the On-the-fly Metaheuristic Design problem.

References

References I

Bartz-Beielstein, T., Lasarczyk, C., and Preuss, M. (2005).

Sequential Parameter Optimization.

In *IEEE Congress on Evolutionary Computation*, pages 773–780.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002).

A Racing Algorithm for Configuring Metaheuristics.

In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18.

Birattari, M., Yuan, Z., Balaprakash, P., and Stützle, T. (2010).

F-Race and Iterated F-Race: An Overview.

In Bartz-Beielstein, T., Chiarandini, M., Paquete, L., and Preuss, M., editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer Berlin Heidelberg.

Hutter, F., Hoos, H. H., and Stützle, T. (2007).

Automatic Algorithm Configuration based on Local Search.

In *Conference on Artificial Intelligence*, pages 1152–1157.

References II

Montero, E. and Riff, M.-C. (2014).

Towards a Method for Automatic Algorithm Configuration: A Design Evaluation Using Tuners.

In Bartz-Beielstein, T., Branke, J., Filipič, B., and Smith, J., editors, *Parallel Problem Solving from Nature - PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 90–99. Springer International Publishing.

Montero, E. and Riff, M.-C. (2015).

RAIS-TTP Revisited to Solve Relaxed Travel Tournament Problem.

In *Genetic and Evolutionary Computation Conference*, GECCO '15. ACM.

Montero, E., Riff, M.-C., Pérez-Cáceres, L., and Coello Coello, C. A. (2012).

Are State-of-the-Art Fine-Tuning Algorithms Able to Detect a Dummy Parameter?

In Coello Coello, C. A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., and Pavone, M., editors, *Parallel Problem Solving from Nature - PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 306–315. Springer Berlin Heidelberg.

Nannen, V. and Eiben, A. (2007).

Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters.

In *International Joint Conference for Artificial Intelligence*, pages 975–980.

References III

Pelikan, M. (2008).

Analysis of Estimation of Distribution Algorithms and Genetic Algorithms on NK Landscapes.

In *10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 1033–1040. ACM.

Pérez-Cáceres, L. and Riff, M.-C. (2015).

Solving scheduling tournament problems using a new version of clonalg.

Connection Science, 27(1):5–21.

Pierrard, T. and Coello Coello, C. A. (2012).

A Multi-Objective Artificial Immune System Based on Hypervolume.

In Coello Coello, C. A., Greensmith, J., Krasnogor, N., Liò, P.and Nicosia, G., and Pavone, M., editors, *Artificial Immune Systems*, volume 7597 of *Lecture Notes in Computer Science*, pages 14–27. Springer Berlin Heidelberg.

Riff, M. C. and Montero, E. (2013).

A New Algorithm for Reducing Metaheuristic Design Effort.

In *IEEE Congress on Evolutionary Computation*, pages 3283–3290.

Solnon, C. (2002).

Ants can Solve Constraint Satisfaction Problems.

Evolutionary Computation, IEEE Transactions on, 6(4):347–357.