A Reinforcement Learning approach for the Cross-Domain Heuristic Search Challenge

Luca Di Gaspero, Tommaso Urli

Università degli Studi di Udine
Dipartimento di Ingegneria Elettrica,
Gestionale e Meccanica

Overview

Goals of our work

The goals of our work

- design an intelligent hyper-heuristic to participate in this year's Cross-Domain Heuristic Search Challenge (CHeSC)
- explore **reinforcement learning** [Sutton, 1988] as a way to evolve an intelligent behavior to guide the search

CHeSC 2011

The **CHeSC** [Burke et al., 2011] is

- an algorithm competition
- fostering research in the field of hyper-heuristics [Burke et al., 2010]
- organized this year for the first time by the ASAP research group of the University *of* Nottingham

Main ideas

Main ideas behind CHeSC:

- domain-specific low-level operators (*low-level heuristics* in [Burke et al., 2010]) are given
- domain-independent hyper-heuristic must be developed
- optimization must be cross-domain

The HyFlex framework

The HyFlex [Burke et al., 2009] framework for CHeSC consists in:

- a Java API
- **four** pre-implemented problem domains
- **four** classes of operators
- solution initialization and evaluation facilities

Problem domains

Four problem domains (plus **two**, TSP and VRP, disclosed at the competition day):

- Boolean satisfiability (MAX-SAT)
- One-dimensional Bin Packing
- Permutation Flow Shop
- Personnel Scheduling

with **ten** training instances each and specialized low-level operators.

Low-level operators

Low-level operators in HyFlex:

- **local-search** operators (non-deteriorating)
- **mutation** operators
- **cross-over** operators
- **ruin-recreate** operators

retrievable by index (single) or by family (set). **The hyper-heuristic is not aware of the identity of an operator.**

Reinforcement learning

Reinforcement learning (RL) [Sutton, 1988]

- popular trial-and-error machine learning approach
- learning based on rewards (positive or negative)

Key components:

- the **environment**
- the **reward** function
- the **policy**
- the **learning function**

Environment

The environment represents an agent's interface with the outside.

Main components:

- the possible **actions** an agent can take
- the observable (or estimable) **state** of the environment

The goal is to map each environment state (situation) to the best action for maximizing a long-term utility.

Actions

Actions are represented by pairs

(operator family, intensity of application)

Once an operator family has been chosen, an random operator in the family is applied with the specified **intensity** (HyFlex parameter).

- finer-grained actions slow down learning
- substantial similarities in operators of the same family

Reward

The **reward** represents what is **good** or **bad** for the agent. Computed as $\Delta_{\text{cost function}}$ of the candidate solution before and after applying the action.

Other possible reward functions, e.g. $\Delta_{\text{cost function}}/\text{time}$, relative rewards . . . not significant boost in performance.

States

A **state** represents **what an agent knows** about the environment at a given time, it describes a *situation*.

The kind of reasoning we want our algorithm to do is, e.g.

> *"If recent rewards have been very low (e.g. local minima or a plateau) I could try a ruin-recreate step to get out of the situation."*

States, continued

Design principles and constraints

- **few states**, to support on-line learning within the **timeout**
- **domain-independent** because we need to reason at the hyper-level (plus the domain is hidden by HyFlex)

Note challenging as some relevant problem-independent information (e.g. distance between solutions) **not available** in HyFlex!

States, continued

Our state is a **single integer** number representing the *recent relative reward* received by the agent.

- **Recent** because oldest rewards are discounted (non-stationary)
- **relative** because the reward is considered relative to an **evolving unit of measure** (search phase-independent)

A representation of the current **search trend** (improving, deteriorating).

States, continued

Given a reward r_{t+1} and the previous state s_t, s_{t+1} is computed as:

$$s_{t+1} = \lfloor s_t + \alpha([r_{t+1}/unit] - s_t) \rfloor$$

Where *unit* is the discounted average of the received low rewards (i.e. wrt a *reference reward*, which is a discounted average of **all** received rewards), and *alpha* is step-size parameter which represents the agent's **reactivity**.

Policy and learning function

An $\epsilon - greedy$ **policy**, based on action **value** at the current state, is used to choose which action to take.

The **learning function** keeps moving the value of a (*state*, *action*) pair towards the most recent reward, according to the formula:

$$\pi(s,a)_{k+1} = \pi(s,a)_k + \gamma * (r_k - \pi(s,a)_k)$$

Where γ is a constant **learning rate** parameter (to tackle non-stationarity).

Parameters

Selected by executing an implementation of the **F-Race** [Birattari et al., 2002] tuning framework over **40** training instances.

- $\alpha = 0.1$ (reactivity)
- $\gamma = 0.1$ (learning rate)
- $\epsilon = 0.1$ (probability of taking suboptimal action)

Same framework for select among several **policies**, **learning functions**, **state** and **action** representations, **cooling schedules**, **number of agents**, . . . (**PbO level 2** and **above**).

Parameters, continued

Results of tuning (the higher the better)

State	n	α	γ	Rew.	Policy	ϵ	Score	SAT	Bin P.	Pers. S.	Flow-s.
Trend	n/a	0.1	0.1	Δ_{cost}	ϵ-greedy	0.10	241.00	96.00	64.00	53.50	27.50
Trend	n/a	0.1	0.9	Δ_{cost}	ϵ-greedy	0.01	240.00	98.00	73.00	57.00	12.00
Trend	n/a	0.1	0.1	Δ_{cost}/t	ϵ-greedy	0.10	237.75	92.25	65.00	48.00	32.50
Trend	n/a	0.1	0.9	Δ_{cost}/t	ϵ-greedy	0.10	237.50	65.50	71.00	61.00	40.00
Trend	n/a	0.1	0.1	Δ_{cost}	ϵ-greedy	0.01	236.00	98.00	72.00	48.00	18.00
Trend	n/a	0.1	0.1	Δ_{cost}/t	ϵ-greedy	0.10	235.08	91.58	66.00	35.50	42.00
Trend	n/a	0.1	0.1	Δ_{cost}	ϵ-greedy	0.10	232.00	96.00	64.00	39.00	33.00
Trend	n/a	0.9	0.9	Δ_{cost}/t	ϵ-greedy	0.01	232.00	96.00	76.00	41.50	18.50
Hist.	1	n/a	0.10	Δ_{cost}/t	ϵ-greedy	0.01	231.75	94.25	74.00	57.00	6.50
Hist.	0	n/a	0.10	Δ_{cost}	Softmax	n/a	231.25	93.25	67.00	44.50	26.50
Rand.	n/a	n/a	n/a	n/a	n/a	n/a	51.00	1.00	18.00	19.00	13.00

Figure: Top ten hyper-heuristics developed by the research unit for CHeSC 2011, together with a comparison with a **random walk** (last row).

Results (the **bad** news)

At the CHeSC 2011, our algorithm has been ranked *16*th over *20* participants.

Poor results on problem domains with **long action application time**, competitive results on other problem domains (e.g. SAT, Bin Packing, VRP).

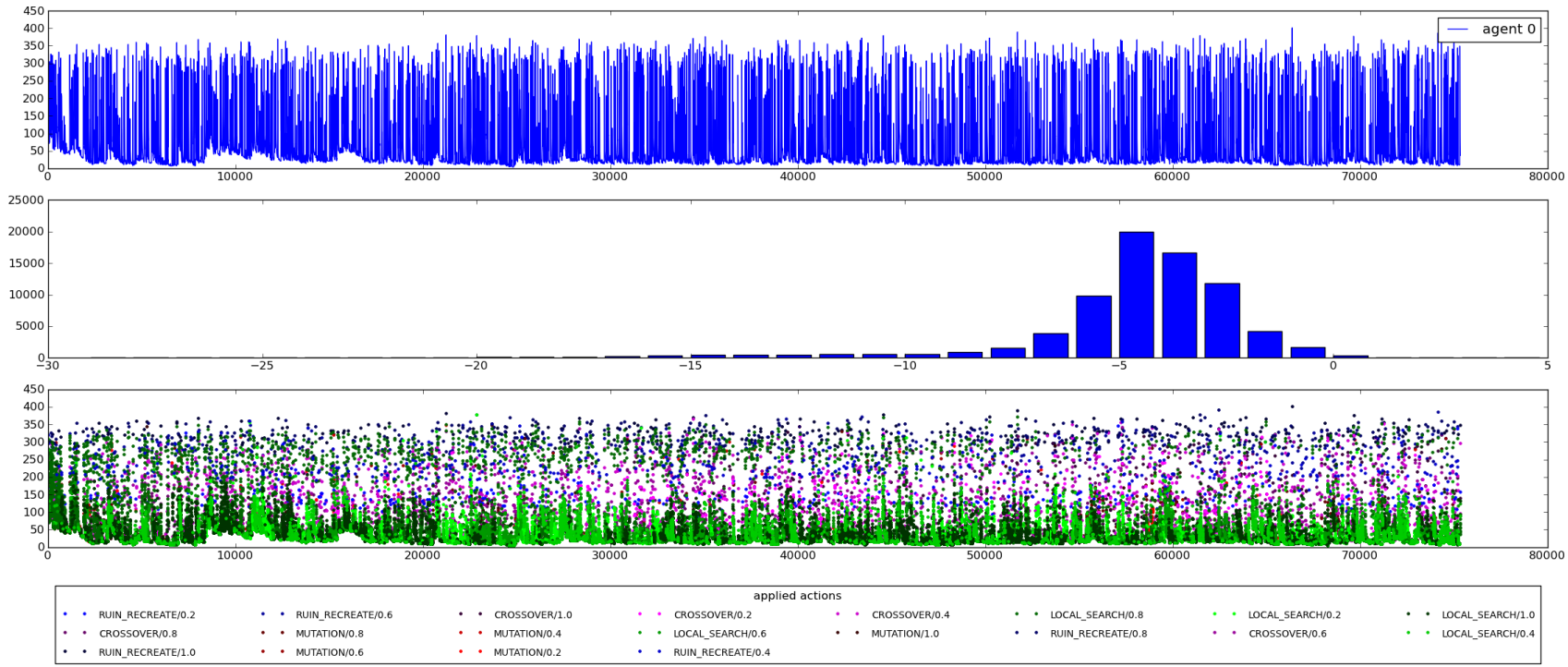
But some interesting emerging patterns on SAT.

Emerging patterns: ILS

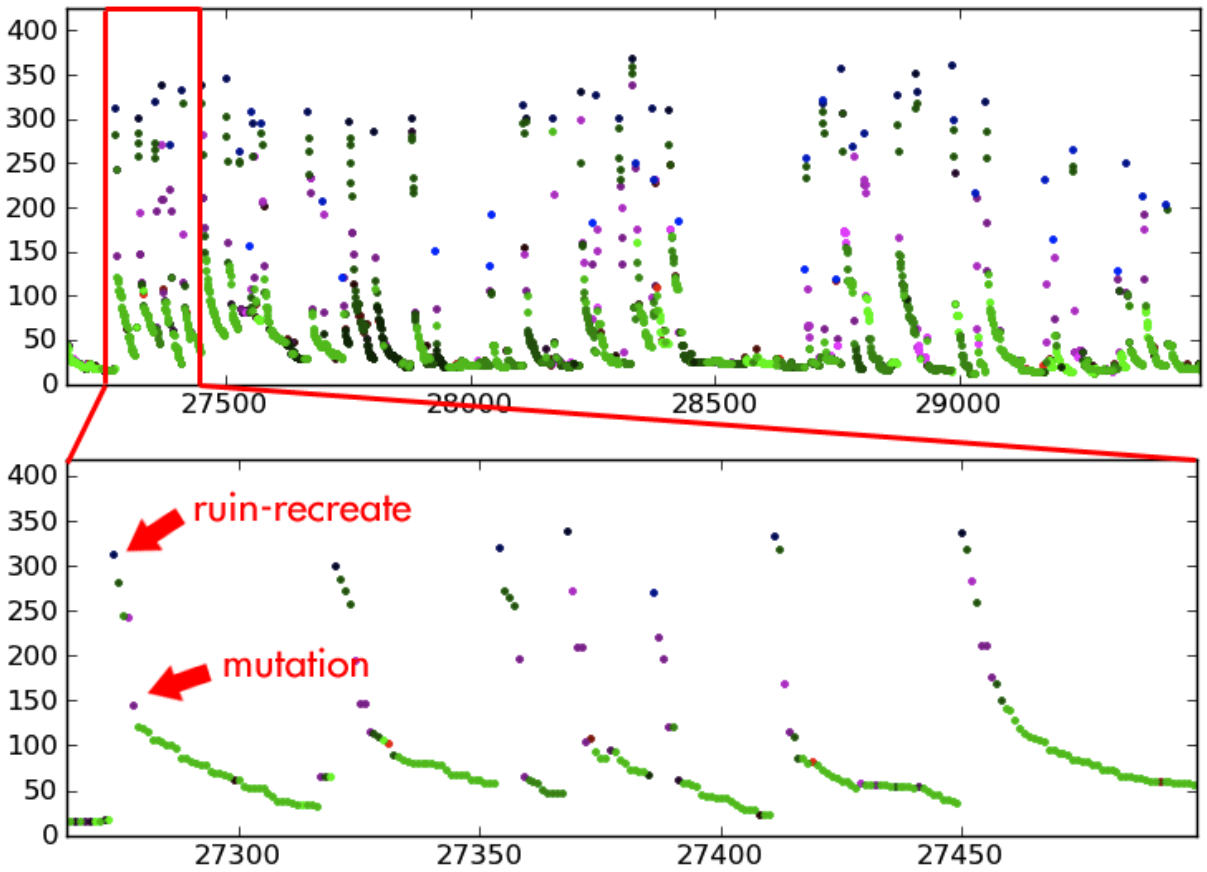
Basic reward and punishment mechanism generates an ILS-like behavior.

Ruin-recreate operators (blue) followed by chain of **local search** operators (green).

Emerging patterns: trace of an agent



Emerging patterns: details of ILS behavior



Byproducts

A framework for generating and running experiments

- generation based on *JSON* **AND/OR parameter** tree
- **parallel execution** over multiple cores (single machine)
- **parameter-indexed persistence** of results with *MongoDB* (document-based DBMS)
- programming by optimization based on *F-Race*
- visualization of algorithm trace through *matplotlib*
- written in **Java** (to integrate with HyFlex)

Future work

Future improvements on this algorithm are

- switch to **function approximation techniques** (e.g. neural nets) for representing the state-action value function (vs. current tabular approach)
- adopt a more **informative state representation**
- introduce **eligibility traces** [Sutton, 1988] to speed-up the learning process
- training on more instances/domains

Questions

Thank you for your attention,
do you have any question?

References I

Birattari, M., Stutzle, T., Paquete, L., and Varrentrapp, K. (2002).
A racing algorithm for configuring metaheuristics.
In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 11–18. Citeseer.

Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., and Vazquez-Rodriguez, J. (2009).
HyFlex: A flexible framework for the design and analysis of hyper-heuristics.
In *Multidisciplinary International Scheduling Conference (MISTA 2009)*, pages 790–797.

References II


Burke, E., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A., and Petrovic, S. (2011).
The Cross-domain Heuristic Search Challenge,ÄìAn International Research Competition.
In *International Conference on Learning and Intelligent Optimization (LION5)*, pages 2–5.


Burke, E., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., and Qu, R. (2010).
Hyper-heuristics: A Survey of the State of the Art.
Technical report.

References III

Sutton, R. S. (1988).
Learning to predict by the methods of temporal differences.
Machine Learning, 3(1):9–44.