A Hyper-heuristic Inspired by Pearl Hunting

C.Y. Chan, Fan Xue, W.H. Ip, and C.F. Cheung

Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hunghom, Kowloon, Hong Kong
{mfcychan,mffxue,mfwhip,mfbenny}@inet.polyu.edu.hk

1 Pearl Hunter: An Inspired Hyper-heuristic

Pearl hunting is a traditional way of diving to retrieve pearl from pearl oysters or to hunt some other sea creatures. In some areas, hunters need to dive and search seafloor repeatedly at several meters depth for pearl oysters. In a search perspective, pearl hunting consists of repeated diversification (to surface and change target area) and intensification (to dive and find pearl oysters). A Pearl Hunter (PHunter) hyper-heuristic is inspired by the pearl hunting, as shown in Fig. 1. Given a problem domain and some low-level heuristics (LLHs), PHunter can group, test, select and organize LLHs for the domain by imitating a rational diver.

PHunter, which executes a repeated "move-dive-move-dive" sequence in the main phase in Fig. 1, is in the Iterated Local Search (ILS) scheme [1] in general. In PHunter, a "surface move" (or move) action involves usually one diversification LLH which is not hill climbing. However, PHunter can try more moves if the new position (solution) is trapped around a "buoy". In other words, a diversification will not be accepted if the new objective value does not meet a low threshold "buoy". In practice, the buoy can be set to the best result of the first iteration.

A "dive" action refers to a sequential execution of hill climbing LLHs. There are two kinds of dives: "snorkeling" and "deep dive" (scuba). Snorkeling involves a short sequence of hill climbing algorithms with a low "depth of search" and stops once an improvement is found. Deep dive iteratively carries out a long sequence with a high "depth of search" until no further improvement can be found. Experience showed that there were different positive coefficients between snorkeling and deep dive in different domains. In a typical "move-dive" iteration, PHunter generates a number ($Num\_of\_snorkeling$) of new solutions and ranks them by snorkeling. Only a few promising (best ranked) solutions can be further processed by deep dives.

PHunter decides a "mode" consisting of a portfolio of grouped moves and a way of diving for a given problem. In fact, the idea of portfolio was proven successful in SAT (Boolean satisfiability) competitions [2]. In the rehearsal run, PHunter employs counters to record how many suboptimal solutions are found by different groups of moves and different dives. The final mode is determined according to the rules obtained by off-line learning. The diving environment is also discovered in the rehearsal run. For example, if the snorkeling and the deep dives always find the same result for every move, the environment is flagged as

```
1 procedure PHunter()
2     test_and_order_dives_and_moves();
3     mode ←rehearsal_run();
4     loop while (terminate_condition_not_met())
5         for each move m in mode.portfolio
6             P ←∅;
7             loop for Num_of_snorkeling times
8                 p ←apply_move_to_pool(m);
9                 loop while (trapped_around_buoy(p))
10                    p ←apply_more_moves(p);
11                end loop
12                p′ ←snorkeling(p, mode.env);
13                P ← P ∪ p′;
14            end loop
15            p* ←select_promising_positions(P);
16            deep_dive(p*, mode.env);
17        end for
18        if (mission_restart_condition_is_met())
19            clear_pool();
20        end if
21    end loop;
22    return BestEverFound;
23 end procedure
```

Fig. 1. Pseudo code of Pearl Hunter

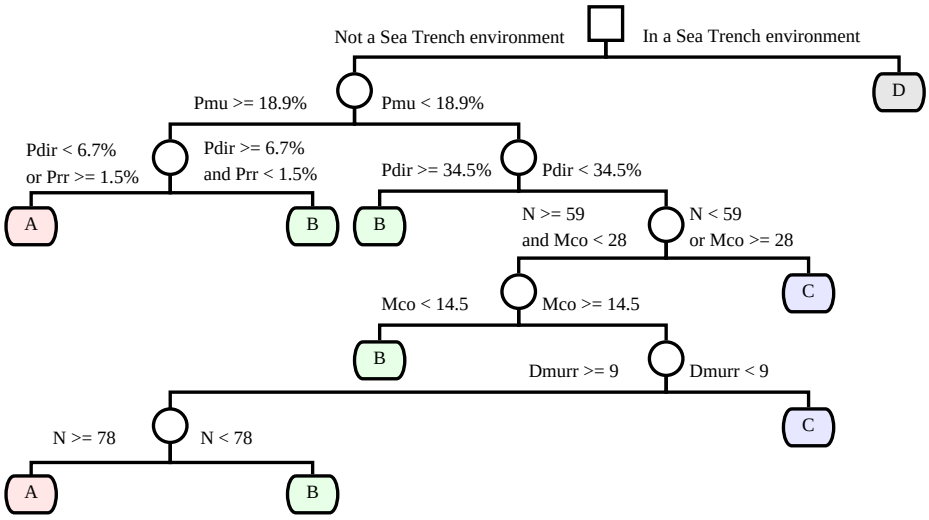
"Shallow Water". In Shallow Water, PHunter simplifies the sequence in snorkeling and disables deep dives. Another environment is "Sea Trench", where at least one hill climbing heuristic consumes too much time (e.g., 3% of overall time) in a single execution. In this case, the depths of search are tuned to lower values and the sequences in snorkeling and deep dives are also simplified.

In practice, a hashed cache can be employed to record courses of deep dives and it is also used as an unwanted (inferior to tabu) list for surface moves at the same time. A restart mechanism can reset the search procedure when the suboptimal solution pool is over-converged or no better solutions are found for a certain amount of time.

2 Implementation and Experiments

PHunter was implemented on a Java cross-domain platform named HyFlex[1] (Hyper-heuristics Flexible framework) [3]. HyFlex provides a random initialization, a set of LLHs in 4 groups (Crossover, Mutation, Ruin-recreate and Local search), two parameters (the "intensity" of mutation and "depth of local

[1] See `http://www.asap.cs.nott.ac.uk/chesc2011/hyflex_description.html`



D_{murr}: Depth of the mission in the Mutation and Ruin-recreate test,
M_{co}: Number of missions completed in the Crossover test,
N: Number of suboptimal solutions found in total,
P_{dir}: Percentage of suboptimal solutions found right after some moves (before any dive),
P_{mu}: Percentage of suboptimal solutions found in iterations started with Mutation moves,
P_{rr}: Percentage of suboptimal solutions found in iterations started with Ruin-recreate moves,

Fig. 2. Decision tree on modes obtained by off-line learning

search"), and a list of easily accessible (but functionally limited) solutions for each problem in each problem domain.

Five portfolios of moves were defined on the 3 groups (Crossover, Mutation and Ruin-recreate) of moves: average calls (A), Crossover emphasized (B), Crossover only (C), average calls with an online pruning (D), Mutation and Ruin-recreate only (E). The portfolio A chooses a move from the 3 groups with the same probability. The portfolio D selects in the same way and eventually prunes some moves according to the history. An off-line classification procedure was carried out to identify the best mode. The decision attributes (counters) were gathered from a 1-minute test on mode C followed by a 1-minute test on mode E. The latter test inherits the solution pool. A decision tree was discovered by the Best-first tree classifier provided by WEKA[2] with default parameters, as shown in Fig. 2, where the mode E was dominated.

Given a set of hill climbing heuristics $\{A, B, C\}$ (ordered by performances) and an initial solution, the result of applying heuristics in order "ABC" is usually different from that in "CBA" in practice. Possible reasons include complex shape of solution space and occasionally inconsistency of local search algorithms. In PHunter, deep dives exploit parallel sequences (such as "A-BA-CBA" and

[2] Version 3.5.6, see `http://www.cs.waikato.ac.nz/ml/weka/`

Table 1. Scores of hyper-heuristics on the HyFlex framework

Domain	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	PHunter	PH_{w/o_Snor}	PH_{Hsiao}
MAX-SAT	41.3	54.8	19.5	5.0	0.0	25.0	38.3	2.0	**91.3**	46.0	67.0
1D Bin-packing	29.0	30.0	53.0	35.0	0.0	23.0	12.0	0.0	**73.0**	63.0	72.0
Personnel Scheduling	**53.0**	50.5	8.0	39.0	42.0	0.0	36.0	18.0	52.0	42.0	49.5
Flow Shop	12.0	3.0	11.0	54.5	5.0	42.5	3.0	37.0	**82.5**	65.0	74.0
Overall	135.3	138.3	91.5	133.5	47.5	90.5	89.3	57.0	**298.8**	216.0	262.5

"*CBA-BA-A*"), swap and repeat until no improvements can be further found. The complex sequences introduce potential redundancy but return better results generally.

Tests have been conducted on 4 problem domains: MAX-SAT, 1D Bin-packing, Personnel Scheduling and Flow Shop, each with 10 difficult instances. The results are shown in Table 1, where PH_{w/o_S} was the PHunter without snorkeling and PH_{Hsiao} was the PHunter that used Hsiao et al.'s local search scheme [4] in deep dive. The results were on average of 10 independent trials. HH1 to HH8 were 8 default hyper-heuristics and their results were provided in HyFlex. The scoring system was the Formula 1 point system provided by HyFlex, greater number meant better. The computation time was benchmarked to be equal to 10 CPU minutes on an Intel P4 3.0GHz CPU.

As shown in Table 1 scores of PHunter were competitive. PHunter won the 4th place overall and the 1st place in the hidden domains out of 20 competitors in the CHeSC 2011 competition[3]. The score of PH_{w/o_Snor} was significantly lower than PHunter's in Table 1. It can be concluded that the snorkeling trial is one of the keys to the success of PHunter. Another key should be the effectiveness of the ILS scheme. The parallel sequences of local search tests in deep dive might also be a reason by comparing the scores of PHunter and PH_{Hsiao}. However, the complex sequence in deep dive is a specified compromise to the LLHs implemented in HyFlex and may not work in other practice.

In fact some results of the tests approximated or had broken the best-known solutions. One exception is the Personnel Scheduling domain. PHunter classified most of the environments as Sea Trench in the domain. So further tests were made, where the computation time was 24 CPU hours and more benchmark problems were included. The results were much satisfied. Especially, PHunter discovered 6 new best-known records, as shown in Table 2. One possible reason was the new "vertical" swap local search was first implemented in an LLH on the HyFlex.

Acknowledgements

The work described in this paper was partially supported by a grant from the Hong Kong Polytechnic University (POLYU5110/10E) and partially supported

[3] See `http://www.asap.cs.nott.ac.uk/chesc2011/results.html`.

Table 2. New best-known solutions found in the Personnel Scheduling domain

Instance	Size (Men * days)	Time (h)	Solution	Previous best-known†	% improved
BCV-A.12.2	12 * 31	24	1,875	1,953	4.0
CHILD-A2	41 * 42	24	1,095	1,111	1.4
ERMGH-B	41 * 42	24	1,355	1,459	7.1
ERRVH-A	51 * 42	24	2,135	2,197	2.8
ERRVH-B	51 * 42	24	3,105	6,859	54.7
MER-A	54 * 42	24	8,814	9,917	11.1

†: Collected from `http://www.cs.nott.ac.uk/~tec/NRP/misc/NRP_Results.xls`.

by a grant from the Department of Industrial and Systems Engineering of The Hong Kong Polytechnic University (No. RP1Z).

References

1. Lourenço, H., Martin, O., Stützle, T.: Iterated Local Search. In Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics. Springer New York. 320–353 (2003)
2. Xu, L., Hutter, F., Hoos, H., and Leyton-Brown, K.: SATzilla: Portfolio-based Algorithm Selection for SAT. Journal of Artificial Intelligence Research. vol. 32 565–606 (2008)
3. Ochoa G., Hyde M., Curtois T., Vazquez-Rodriguez J. A., Walker J., Gendreau M., Kendall G., McCollum B., Parkes A. J., Petrovic S., and Burke E. K.: HyFlex: A Benchmark Framework for Cross-domain Heuristic Search. European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2012), LNCS series, Vol. 7245, Springer. (2012)
4. Hsiao, P.-C., Chiang, T.-C., Fu, L.-C.: A variable neighborhood search-based hyperheuristic for cross-domain optimization problems in CHeSC 2011 competition. Fifty-third Conference of OR Society (OR53), September 6–8, Nottingham, UK (2011)