Genetic Hive HyperHeuristic

Michał Frankiewicz, Tomasz Cichowicz, Maciej Drozdowski,
Grzegorz Pawlak, Filip Rytwiński, and Jacek Wasilewski

Institute of Computing Science,
Poznań University of Technology,
Piotrowo 2, 60-965 Poznań, Poland
Michal.Frankiewicz@student.put.poznan.pl
Tomasz.Cichowicz@student.put.poznan.pl
Maciej.Drozdowski@cs.put.poznan.pl
Grzegorz.Pawlak@cs.put.poznan.pl
Filip.Rytwinski@student.put.poznan.pl
Jacek.Wasilewski@student.put.poznan.pl

1 Algorithm Description

This algorithm consists in parallel search of the solution space using evolving sequences of low level heuristics (LLH). It is inspired by the behaviour of bees searching for food. At the given moment in time part of the bee population actively searches food locations, while the remaining bees stay at the hive. In time some bees leave the search locations and return to the hive. And vice versa, some of the bees previously in the hive set out exploring the locations. The successful bees have bigger chances to proliferate. In the algorithm bees correspond to the sequences of LLH. We will call them agents. Search locations correspond with the current problem solutions. The agents (bees) in the hive remain passive, while the agents out of the hive attempt improving the current solutions. Thus, this algorithm is a combination of evolutionary approach and simulation of agent colony searching for resources.

Let us denote by S the set of all agents, by D the set of active agents (out of the hive), and by $X \subseteq D$ the set of agents continuing their search in their locations.

In the initial phase of the algorithm a set of random agents (bees) S, and a set of random solutions (search locations) are created. The sizes of both sets are algorithm control parameters. Agents are randomly selected to set D and randomly assigned to the search locations.

The algorithm consists of several steps repeated iteratively until exploiting the time limit. Firstly, the agents (sequences of LLH) from set D are applied in the search locations (the solutions) to which they were assigned. In the evaluation each agent obtains a score which measures relative solution improvement.

In the following step, a fraction of the best agents is elected to set X and remain in their locations. The agents in set $D - X$ are moved to the hive S. Size of set X is algorithm control parameter. The abandoned search slots are assigned new agents constructed in the following way.

Set Y of invariant passive agents is selected. The remaining part is constructed by crossing-over and mutation of agents from set X. Crossing over of the agents (sequences of LLH) consists in the exchange of sequence prefixes. Agents are elected to the crossing over by roulette wheel method with probabilities proportional to the agent score. Mutation changes randomly selected member of the agent LLH sequence. Agents constructed in the above way are assigned randomly to the search locations of the agents from set X.

The algorithm is controlled by the parameters:

1. number of search locations $|D|$,
2. population size - the total number of agents $|S|$,
3. size of the agent - number of LLHs in agent sequences,
4. fraction of the agents remaining in their search locations, $|X|/|D|$,
5. fraction of the population replaced by the offspring,
6. probability of mutation.

2 Pseudocode

```
HiveAgent : record
begin
   int SearchLocation;
   int [1..AgentSize] LLH;
   double Score;
end

S = Randomly generated set of HiveAgents;
M[1..|D|] = Initial SearchLocations;

Assign randomly HiveAgents to search locations M[1..|D|],
and put them into set D;

while(time < TimeLimit)
begin
   Evaluate HiveAgents in set D;

   Select set X of best HiveAgents remaining at their SearchLocations;

   Select set Y from set S-X of HiveAgents invariant in this iteration;

   Construct set P of offspring such that |P|=|((S-X)-Y)|;

   Set S := X sum P sum Y;

   Randomly assign HiveAgents from set S-X to |D-X| free SearchLocations;
end
```