hLCGA: A Hybrid Competitive Coevolutionary Genetic Algorithm

Grégoire Danoy, Pascal Bouvry, Tomy Martins
University of Luxembourg
Faculty of Science, Communication and Technology
6 rue Coudenhove Kalergi
L-1359 Luxembourg, Luxembourg
Email: \{gregoire.danoy, pascal.bouvry, tomy.martins\}@uni.lu

Abstract—We introduce in this article a new hybrid coevolutionary algorithm called hLCGA (hybrid Loosely Coupled Genetic Algorithm) that consists in combining a competitive coevolutionary genetic algorithm and a local search algorithm. We apply it to the Rosenbrock function optimization problem and compare the results of five hybrid variants to the original LCGA. We show the advantages of hybridizing a coevolutionary algorithm with local search algorithms in terms of solution quality and convergence speed.

I. INTRODUCTION

The use of evolutionary computation (EC) techniques to evolve solutions of both theoretical and real-life problems has seen a dramatic increase in popularity and success over last decade. The most popular and widely applied EC technique was a sequential GA [1] which computational scheme is based on a single population of individuals representing a single species. Development of parallel machines stimulated parallelization of a sequential GA and resulted in two parallel EC techniques known respectively as island model and diffusion model (see, e.g. [2]). Both models, which are widely used today, have been exploring a notion of a single species, and individuals representing this species live in different subpopulations.

While these techniques are very effective in many applications, new more difficult problems were set. These problems (e.g. modelling economic phenomena such as a market) are in their nature distributed, i.e. they can be seen as a number of independent interacting entities with their own goals, where a global behavior can be observed as the result of interactions. To meet these new requirements, researches in the area of EC were looking for new more powerful paradigms of natural processing. In the result coevolutionary algorithms based on modelling phenomena of coexistence of several species emerged [3] as a very promising area of EC.

Hybridization in evolutionary computations aims at combining different approaches with evolutionary algorithms (EA) in order to take advantage of each approach. We study in this article a hybrid competitive coevolutionary algorithm called hLCGA (hybrid Loosely Coupled Genetic Algorithm) and apply it to a classical test function: Rosenbrock. The hybridization of LCGA relies on the addition of local search to the evolutionary process. The main objective of our approach is to keep the highly scalable solution provided by LCGA and increase its convergence speed and solutions quality by combining it with local search algorithms.

In the coming section, we introduce the concept of hybrid coevolutionary algorithms. In section III we provide a detailed description of LCGA and of the newly created hybrid LCGA. Then in section IV we describe the experiments done on the Rosenbrock test function and discuss of the obtained results. The last section contains our conclusions and perspectives.

II. HYBRID COEVOLUTIONARY ALGORITHMS

As for the "classical" genetic algorithm, the concept of coevolutionary algorithms comes from biological observations [4]. Indeed, the nature is composed of several species that coevolve and thus instead of evolving a population of similar individuals representing a global solution (like in classical Genetic Algorithms), we consider the coevolution of subpopulations of individuals representing specific parts of the global solution. Coevolutionary algorithms is a very active research field that finds its origin with Hills who first introduced the predator-prey paradigm in 1990 for a sorting network problem [3], followed by Paredis in 1994 [5] who used the same algorithm in the context of a CSP (Constraint Satisfaction Problem) and a neural network optimization problem. Since then, a number of coevolutionary algorithms have been presented, either cooperative like CC GA (Cooperative Coevolutionary Genetic Algorithm) from DeJong [6] or competitive like LCGA (Loosely Coupled Genetic Algorithm) from Seredynski [7].

Hybridization of GAs for optimization purposes has been widely studied in the last years, combining the capacity of GAs to explore huge search spaces and find good regions of solutions with the exploitation power of local search algorithms. Hybrid coevolutionary algorithms are a new and promising alternative but contrary to hybrid GAs they are still quite unexplored. Indeed, only Son and Baldwick in [8] recently proposed a hybrid coevolutionary algorithm for Nash equilibrium search in games with local optima. They have incorporated a local hill-climbing algorithm to the basic coevolutionary algorithm, which is in fact a CCGA, so as to speed up convergence and fine tune control variables. To this end, they introduced the concept of "best rival matching and fine tuning" meaning that the chosen individual of one
population is matched against the best strategies of the other populations in each generation and optimized using the hill climber. As experiment, they applied their hybrid algorithm on a transmission-constrained electricity market problem, showing that the hybrid version successfully avoided NE traps.

III. FROM LCGA TO HLCGA

The Loosely Coupled Genetic Algorithm (LCGA) [7] is a medium-level parallel and distributed coevolutionary algorithm exploring a paradigm of competitive coevolution motivated by non-cooperative models of game theory.

For an optimization problem described by some function (a global criterion) of N variables, local chromosome structures are defined for each variable and local subpopulations are created for each of them. As opposed to known sequential and parallel EAs, the LCGA is assumed to work in a distributed environment described by locally defined functions. A problem to be solved is first analyzed in terms of possible decomposition and relations between subcomponents that are expressed by a communication graph Gecom, aka graph of interaction. The objectives of this function decomposition and of the definition of the interaction graph are to minimize communications while still ensuring that the fact of reaching local optima for all different players (being a Nash equilibrium point) still leads to a global optimum of the initial function. This process has still to be done manually by taking into account information on the internal structure of the cost function, i.e. of the problem. The LCGA approach has been successfully applied on various problems including optimization of hard mathematical functions, multiprocessor mapping and scheduling. LCGA can be specified in the following way:

```
Algorithm 1: LCGA

    gen = 0
    foreach player s do
        Pop_s(gen) = randomly initialized population
        evaluate local fitness of each individual i in Pop_s(gen) : u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
    end
    while termination condition = false do
        gen = gen + 1
        foreach player s do
            select Pop_s(gen) from Pop_s(gen - 1) based on fitness
            apply genetic operators to Pop_s(gen)
            evaluate local fitness of each individual i in Pop_s(gen) : u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
        end
    end
```

After initializing subpopulations, corresponding sequences of operations are performed in parallel for each subpopulation, and repeated in each generation. For each individual in a subpopulation a number of n_i (n_i-number of neighbors of subpopulations P(\cdot)) of random tags is assigned, and copies of individuals corresponding to these tags are sent to neighbor subpopulations, according to the interaction graph. Individuals in a subpopulation are matched with ones that arrived upon request from the neighbor subpopulations. Local fitness function of individuals from subpopulations is evaluated on the basis of their values and values of arrived tagged copies of individuals. Next, standard GA operators are applied locally in subpopulations. Coevolving this way subpopulations compete to maximize their local functions. The process of local maximization is constrained by neighbor subpopulations, sharing the same variables. A final performance of the LCGA operated in a distributed environment is evaluated by some global criterion, usually as a sum of local function values in an equilibrium point. This global criterion is typically unknown for subpopulations (except the case when Gecom is a fully connected graph), which evolve with their local criteria.

Although it is possible to make a genetic algorithm hybrid in different ways, latest published articles have shown that combining genetic algorithms with local search algorithms are one of the best approaches for improving the results. For this reason we chose to hybridize LCGA with various local search algorithms that are described below:

- Steepest Ascent Hill Climbing (SAHC): systematically flips each single bit of the chromosome and records all the obtained fitness values. The resulting chromosome with the highest fitness is kept if its fitness value is better than the fitness value of the initial chromosome. If no modified chromosome gives a better result, the algorithm stops.
- Next Ascent Hill Climbing (NAHC): systematically flips each bit from left to right until the resulting fitness increases. If the fitness increased then the flipped bit is kept otherwise it is flipped back. If there is no improvement once at the end of the chromosome, the algorithms stops.
- Random Bit Climbing (RBC) [9][10]: similar to NAHC, except that there is no left to right iteration. Instead a random permutation is generated to determine the order in which bits flips are tested. After flipping every bit in the initial solution string, a new random sequence is chosen for testing the bits and the bit climber again checks every bit for an improvement. If the bit climber has tested every bit and no improvement is found, a local optimum has been reached.
- Dynamic Hill Climbing (DHC): introduced by Yuret in [11], it is based on the following main key heuristics: adjusting the size of probing steps to suit the local nature of the terrain, shrinking when probes do poorly and growing when probes do well and keeping track of the directions of recent successes, so as to probe preferentially in the direction of most rapid ascent.
- Tabu Search (TS) [12]: The tabu search algorithm is built around two important aspects, the tabu list that memorizes the last forbidden moves and the aspiration criteria that allows a tabu status of a tabu move to be
overwritten. In case tabu is combined with GA, every individual in the population maintains a tabu list. The tabu list is kept from one generation to the other until the individual is replaced by offspring.

Below is the hybrid LCGA algorithm showing where the local search has been incorporated:

```
1. gen = 0
2. foreach player_s do
   3. Pop_s(gen) = randomly initialized population
   4. evaluate local fitness of each individual i in Pop_s(gen)
   5. Pop_s(gen) = u_i(s_0, s_1, ..., s_i, ..., s_{N-1})
6. end
7. while termination condition = false do
   8. gen = gen + 1
   9. foreach player_s do
      10. select Pop_s(gen) from Pop_s(gen - 1) based on fitness
      11. apply genetic operators to Pop_s(gen)
      12. evaluate local fitness of each individual i in Pop_s(gen)
      13. apply local search on a percentage of Pop_s(gen)
   14. end
15. end

Algorithm 2: hLCGA
```

Coming with the local search addition, some new parameters can be fine-tuned: the exchange rate fixing the proportion of the population that will be optimized by the local search (it is also possible to apply local search only on the current best individual) and the choice between a restricted or a complete search. If restricted, the local search algorithm will stop after the first improvement and if complete the local search algorithm will be fully executed.

IV. EXPERIMENTATIONS

hLCGA has been implemented in the Dafo framework currently developed at University of Luxembourg [13]. Dafo is a multi-agent framework dedicated to decomposable function optimization that allows to easily use either cooperative or competitive coevolutionary genetic algorithms. Modelling coevolutionary GAs with a multi-agent system makes explicit the decomposition and resolution strategy (i.e. the interaction graph) by using organizational models explicitly representing the roles and the interactions that are allowed for each agent. We consider the opportunity to embed our players into software agent that is a convenient and elegant way to benefit from existing multi-agent platforms [14] that exempt from writing low-level agent interaction behaviors and ensures the deployment and the distribution of the algorithms.

We have experimented the different hLCGA variants and compared them to the "basic" LCGA on a classical function optimization problem known to be very hard: the Rosenbrock function that is part of De Jong’s five function test suite [15]. The Rosenbrock’s function is a continuous and unimodal function:

\[
f_2(x) = \sum_{i=1}^{n} \left(100 \left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2\right); x \in \mathbb{R}^n, \quad (1)
\]

with \(-2.12 \leq x_i \leq 2.12\), a global minimum \(f_2(x^*) = 0\) at \(x^* = (1, 1, \ldots, 1)\). This global optimum is inside a long, narrow, parabolic shaped flat valley. Finding the valley is trivial, however converging to the global optimum is difficult.

Using (h)LCGA, we consider the problem of minimizing the Rosenbrock’s function as a problem of seeking a minimum in a distributed fashion. We use a multi-agent system, within our framework, with a game-theoretic model of interaction among agents as shown in the interaction graph represented in Fig. 1. For LCGA, each agent, that we will call evoAgent,

\[
f'(x_i, x_j)
\]

optimizes its locally defined function which depends only on its \(x_i\) and the \(x_i + 1\) of its neighbor using a Simple GA (SGA).

\[
f_2'(x_i, x_{i+1}) = 100 \left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2; \quad (2)
\]

This way, each evoAgent runs one subpopulation and individuals in one subpopulation code solution for a variable \(x_i\). For hLCGA a second type of agent is additionally used, localSearchAgent, that will run one of the previously mentioned local search algorithms as illustrated in Fig. 2. In each generation the evoAgent will send a predefined percentage of its population to the localSearchAgent that will run one chosen local search algorithm and sends back the optimized individuals.

Fig. 3 and 4 show the results obtained with LCGA and the five versions of hLCGA for the Rosenbrock problem size of \(n = 10\). The following parameters were set for all the algorithms: sub-populations size was equal to 100, 16 bits binary representation, two-point crossover with \(p_m = 0.8\) (crossover probability) and bit flip mutation with \(p_m = 0.03\) (mutation probability).

Experiments have been conducted with the following local search parameters: exchange rate = 0.35 for hLCGA with SAHC, NAHC, RBC, DHC and 0.03 for Tabu Search. Another strategy exchanging only the best individual has also been
tested. Each exchanged strategy (best individual and population rate) was experimented with both restricted and complete local search.

The results presented hereafter were obtained using the population rate exchange strategy combined with complete local search which is the combination that gave the best results. The averaged best of generation for each algorithm over 30 experiments is presented in fig. 3 and detailed in fig.4.

It clearly appears that for such a continuous and unimodal problem all hLCGAs outperform LCGA both in terms of convergence speed and best result found, the overall best being LCGA-TS that converges the faster to the global optimum (i.e. $f(x)=0$). LCGA-SAHC and LCGA-RBC are the only ones that do not converge to this global optimum. As expected, the drawback is the additional computational time required for the local search algorithms execution. Indeed, when it takes 3 seconds for LCGA to perform one experiment, it increases up to 27 seconds in the worst case for LCGA-TS. Taking this parameter into consideration, LCGA-RBC is the fastest with 4 seconds but as previously mentioned it gets stuck in a local optimum, consequently LCGA-NAHC becomes the best choice since it also reaches the global optimum and takes 5 seconds for one experiment.

V. Conclusion

This article introduced a new hybrid coevolutionary algorithm named hLCGA that consists in combining a competitive coevolutionary genetic algorithm with a local search algorithm. This hybrid algorithm was implemented as a multi-agent system within the Dafo framework and some first experiments allowed to compare five variants of hLCGA (using SAHC, NAHC, RBC, DHC and TS) to the original LCGA on a classical test function, i.e. Rosenbrock. The presented results indicate that all hybrid versions outperform LCGA as well as in terms of solution quality as in terms of convergence speed. It also allowed to quantitatively evaluate the overhead in computational time due to the properties of each local search algorithm. Current work now consists in experimenting hLCGA on a real-business problem called ICP (Inventory Control Parameter) which is an inventory management optimization problem. Its performance will be compared to other algorithms such as SGA, CCGA and LCGA.

References