Nat Comput
DOI 10.1007/s11047-012-9360-7

Dealing with hardware heterogeneity: a new parallel search model

Julián Domínguez · Enrique Alba

Abstract In this article we present Ethane, a parallel heterogeneous metaheuristic model specifically designed for its execution on heterogeneous hardware environments. With Ethane we propose a hybrid parallel search algorithm inspired in the structure of the chemical compound of the same name, implementing a heterogeneous island model based in the structure of the chemical bonds of the ethane compound. Here we also shape a schema for describing a complete family of parallel heterogeneous metaheuristics inspired by the structure of hydrocarbons in nature, HydroCM (HydroCarbon inspired Metaheuristics), establishing a resemblance between atoms and computers, and between chemical bonds and communication links. Our goal is to gracefully match computers of different computing power to algorithms of different behavior (genetic algorithm and simulated annealing in this study), all them collaborating to solve the same problem. In addition to the nice natural metaphor we will show that Ethane, though simple, can solve search problems in a faster and more robust way than well-known panmictic and distributed algorithms very popular in the literature, as well as can achieve a better exploration/exploitation balance during the search process.

Keywords Parallel · Metaheuristics · Heterogeneous · Heterogeneous hardware · Hybrid · Simulated annealing · Genetic algorithm · Ethane · HydroCM

J. Domínguez (✉) · E. Alba
Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain
e-mail: julian@lcc.uma.es

E. Alba
e-mail: eat@lcc.uma.es

1 Introduction

Metaheuristics are an important branch of research since they provide a fast an efficient way for solving problems. In many cases, parallelism is necessary, not only to reduce the computation time, but to enhance the quality of the solutions obtained. Many parallel models exist, both for local search methods (LSMs) and evolutionary algorithms (EAs), and even parallel heterogeneous models combining both methods are present in the literature (Alba 2005b; Alba et al. 2004).

In a modern lab, it is very common the coexistence of many different hardware architectures. It has been proven that such heterogeneous resources can also be used efficiently to solve optimization problems with standard parallel algorithms (Alba et al. 2002; Salto and Alba 2012; Salto et al. 2011), but there exist few works about the design of specific parallel models for an heterogeneous environment.

In this paper we propose a heterogeneous parallel search algorithm designed for its execution in a heterogeneous platform. We will also present a draft of a general model for describing a family of heterogeneous metaheuristics specifically designed for its execution in heterogeneous hardware environments, being inspired in the structure of the hydrocarbons that can be found in nature.

Our contribution is not only methodological, but we also have carried out an analysis in order to study the behavior of our proposal. For our analysis, we have implemented two versions of the algorithm making use of two well-known metaheuristics: steady state genetic algorithm (ssGA) and simulated annealing (SA). We have compared our proposal against the panmictic versions of these algorithms as well as against a unidirectional ring of ssGA islands and another one composed of SA islands, both executed on the same hardware infrastructure. Our results show that the running times of our proposal are faster in all the studied cases and even with a more robust behavior than the reference ssGA and SA rings. We can also see how our proposal is able to maintain a better population diversity than the ssGA ring, thus, Ethane has shown a better exploitation/exploration balance according to our analysis.

This paper is organized as follows. The next section (Sect. 2) provides a brief review of decentralized and parallel metaheuristics. Section 3 explains the proposed algorithm and the model that arises from its chemical inspiration. In Sect. 4 we describe some common performance measures which are used in this work. Section 5 contains the problems, parameters and infrastructure used in our study. The analysis of the tests is discussed in Sects. 6 and 7. Eventually, concluding remarks and future research lines are shown in Sect. 8.

2 Decentralized, heterogeneous and hybrid parallel metaheuristics

In this section we include a quick review on the existing implementations of decentralized and parallel metaheuristics, as well as on heterogeneity. We also include a description of the metaheuristics used in our heterogeneous algorithms and how they qualify as hybrid metaheuristics.

Many parallel implementations exist for different groups of metaheuristics. We will focus in two of the more common families of metaheuristics: EAs and local search metaheuristics (LSMs). On the one hand, EAs are population based methods, where a random population is created and further enhanced through a nature-like evolution process. On the other hand, only one candidate solution is used in LSMs, and it is enhanced by moving through its neighborhood replacing the candidate solution by another one, usually one with a better quality. EAs commonly provide a good exploration of the search space, so they are also called exploration-oriented methods. On the contrary, LSMs allow to find a local optima solution and subsequently they are called exploitation-oriented methods. Many different parallel models have been proposed for each method, and here we present the more representative ones.

2.1 Parallel EA models

A panmictic EA applies its stochastic operators over a single population, which makes them easily parallelizable. A first strategy for its parallelization is the use of a master–slave approach where evaluations are performed in parallel but the population, unless divided, is treated as a whole, maintaining its panmictic behavior. This could be interesting for many tasks, but it does not offer the benefits of a structured population. Therefore, we are going to focus in structured populations, which leads to a distinction: cellular versus distributed EAs (Alba 2005a) (Fig. 1).

- *Distributed EAs (dEA)* In the case of distributed EAs, the population is divided into a number of islands that run an isolated instance of the EA (Fig. 1b). Although there is not a single population the sub-populations are not completely isolated: some individuals are sent from one population to another following a migration scheme. There usually exists only a few sub-algorithms, loosely coupled among them.
- *Cellular EAs (cEA)* In the cellular model, there exists only one population which is structured into neighborhoods, so that an individual can only interact with the individuals inside its neighborhood (Fig. 1c). Different neighborhood structures can lead to a different behavior. In a cellular model there exists a large number of overlapped sub-algorithms and they are tightly coupled (Alba and Dorronsoro 2008).

2.2 Parallel LSM models

Many different parallel models have been proposed for LSMs, but there exist three models that are widely extended in the literature: parallel multistart model, parallel moves model, and move acceleration model (Fig. 2).

- *Parallel multistart model* In this model, several independent instances of the LSM are launched simultaneously (Fig. 2a). They can exchange individuals following a migration scheme. This model can usually compute better and more robust solutions than the panmictic version.
- *Parallel moves model* This model is a kind of master–slave model where the master runs a sequential LSM but, at the beginning of each iteration, the current solution is distributed among all the slaves (Fig. 2b). The slaves perform a move and return the candidate solution to the master, which selects one of them. This model does not alter the behavior of the sequential algorithm.
- *Move acceleration model* The quality of each candidate solution is evaluated in a parallel centralized way (Fig. 2c). It is useful when the evaluation function can be itself parallelized. The move acceleration model does not alter the behavior of the sequential algorithm.

In both, EAs and LSMs parallel models, each sub-algorithm includes a phase for communication with a neighborhood according to some topology. This communication can be

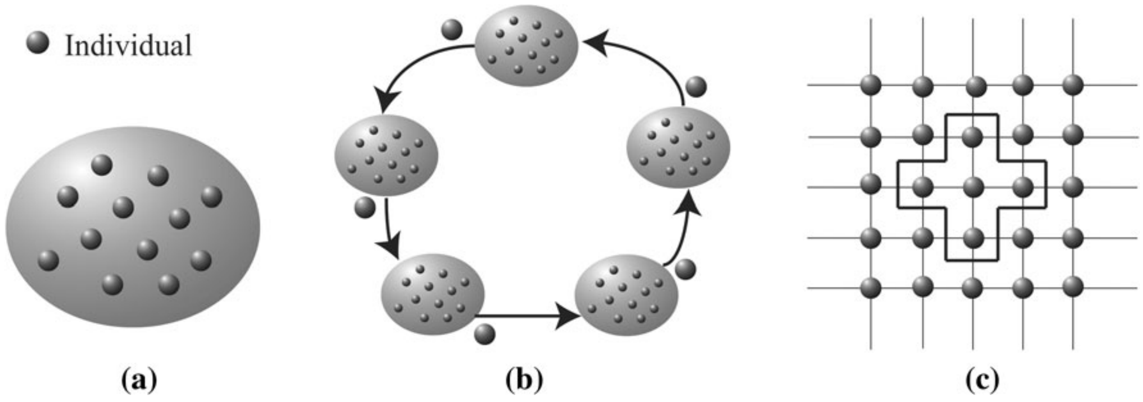


Fig. 1 A panmictic EA (**a**), and two structured EAs: distributed (**b**) and cellular (**c**)

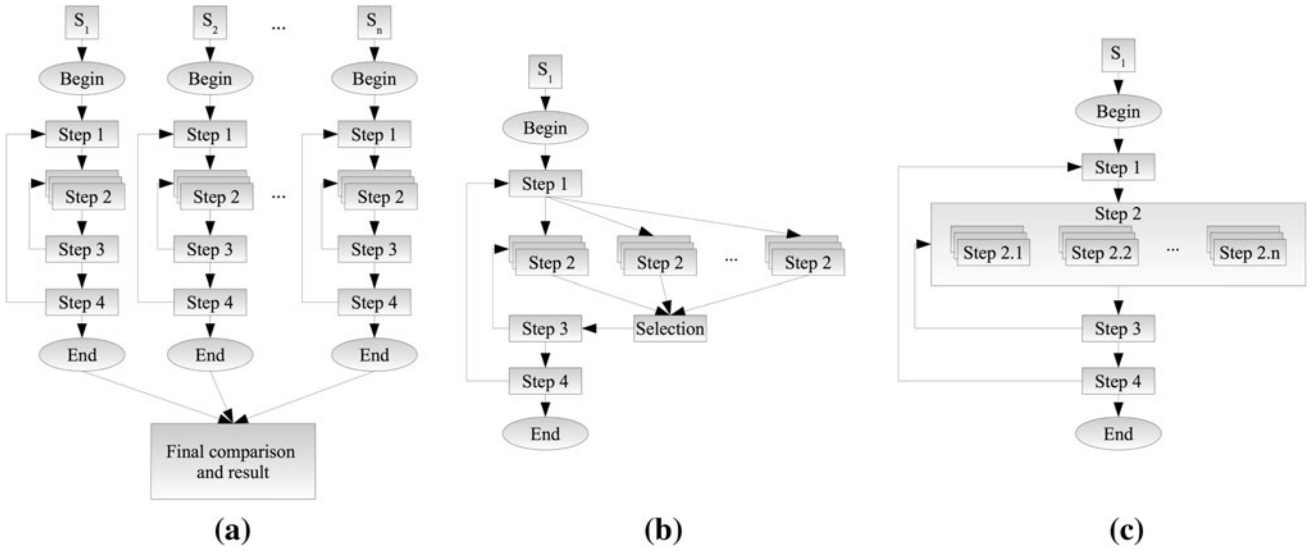


Fig. 2 Parallel multistart model (**a**), parallel moves model (**b**), and move acceleration model (**c**)

carried out in a synchronous or asynchronous manner. Many works have found advantages in using an asynchronous execution model (Alba and Troya 2001; Crainic and Toulouse 2003). Additionally, asynchronism is essential in our study because of the heterogeneous hardware, which could easily produce bottlenecks, so our communications will be carried out in an asynchronous way.

2.3 Achieving the heterogeneity

In the models presented above, all the sub-algorithms share the same search features. But we could modify the behavior of a parallel metaheuristic by changing the search features between sub-algorithms, obtaining a globally heterogeneous hybrid metaheuristic. Also the hardware being used to run the algorithm could be homogeneous or heterogeneous, so we have not to be confused between the hardware platform heterogeneity and the heterogeneous software model. Parallel heterogeneous metaheuristics can be classified into four levels depending on the source of heterogeneity (Alba 2005a):

- *Parameter level* At this level, the same algorithm is used in each node, but the configuration parameters are different in one or more of them.
- *Operator level* At operator level, heterogeneity is achieved by using different mechanisms for exploring the search space, such as different operators.
- *Solution level* Heterogeneity is obtained using a different encoding for the solutions in each component.
- *Algorithm level* At this level, each component can run a different algorithm. This level is the most widely used.

Here we present an algorithm level parallel heterogeneous metaheuristic which is later run in heterogeneous hardware. This solver is based in two different methods. We have chosen one method of each of the two well-known families, LSMs and EAs, in order to obtain a good balance between exploitation and exploration. The used methods are a genetic algorithm (GA) and a SA.

Algorithm 1 Standard Genetic Algorithm

```
Generate($P(0)$);
Evaluate($P(0)$);
t := 0;
while not stop_condition($P(t)$) do
    $P'(t)$ := Selection($P(t)$);
    $P''(t)$ := Recombination($P'(t)$);
    $P''(t)$ := Mutation($P''(t)$);
    Evaluate($P'''(t)$);
    $P(t+1)$ := Replace($P(t)$,$P'''(t)$);
    t := t+1;
end while
```

GAs are one of the more popular EAs present in the literature. In Algorithm 1 we can see an outline of a panmictic GA. A GA starts by randomly generating an initial population $P(0)$, with each individual encoding a candidate solution for the problem and its associated fitness value. At each iteration, a new population $P'''(t)$ is generated using simple stochastic operators, leading the population towards regions with better fitness values.

In our algorithm, we have actually used a special variant of the generic GA called ssGA (Syswerda 1991). The difference between a common generational GA and a ssGA is the replacement policy: while in a generational GA a full new population replaces de old one, in a ssGA only a few individuals, usually one here, are generated at each iteration and merged with the existing population.

Algorithm 2 Standard Simulated Annealing

```
Generate($S$);
Evaluate($S$);
Initialize($T_0$);
k := 0;
while not stop_condition($S$) do
    $S'$ := Generate($S$,$T_k$);
    if Accept($S$,$S'$,$T_k$) then
        $S$ := $S'$;
    end if
    $T_{k+1}$:= Update($T_k$);
    k := k+1;
end while
```

Because of its easy utilization SA has become one of the most popular LSMs. SA is a stochastic algorithm which explores the search space using a hill-climbing process. A panmictic SA is outlined in Algorithm 2. SA starts with a randomly generated solution S. At each step, a new candidate solution S' is generated. If the fitness value of S' is better or equal than the old value, S' is accepted and replaces S. As the temperature T_k decreases, the probability of accepting a lower quality solution S' decays exponentially towards zero according to the Boltzmann probability distribution. The temperature is progressively decreased following an annealing schedule.

Based on the classic SA, many different versions have been implemented by using a different annealing schedule. In our algorithm we have used the new simulated annealing (Yao 1995), which uses a *very fast* annealing schedule.

2.4 Classifying hybrid metaheuristics

Attending to the classification proposed in Talbi (2002) (Fig. 3), we can classify a hybrid metaheuristic attending to its structure (hierarchical) or to the features of the algorithms involved in the hybrid (flat). Four classes are derived from the hierarchical taxonomy:

- *Low-level relay hybrid (LRH)* This class of hybrids represents algorithms in which a given metaheuristic is embedded into a single-solution algorithm. We can find some examples of LRH in the literature (Aarts and Verhoeven 1997; Martin et al. 1992).
- *Low-level teamwork hybrid (LTH)* This class comprises combinations of metaheuristics with strong exploring capabilities (like most EAs) with exploitation-oriented metaheuristics (most single-solution metaheuristics). Usually, exploitation-oriented methods replace or extend genetic operator such as mutation or crossover. There are numerous examples of this strategy, for example Chen and Flann (1994), Fleurant and Ferland (1996), and Lozano et al. (2004).
- *High-level relay hybrid (HRH)* In this class of algorithms, self-contained metaheuristics are executed in a sequence. In HRH, an algorithm is used for improving the results obtained by another one. Many authors have used this idea (Mahfoud and Goldberg 1995; Talbi et al. 1994).
- *High-level teamwork hybrid (HTH)* Self-contained algorithms perform a search in parallel, and cooperating to find an optimum. This model has been widely used in the literature (De Falco et al. 1994; Voigt et al. 1990).

As to the flat classification, we can distinguish between:

- *Homogeneous/heterogeneous* In homogeneous hybrids, all the combined algorithms use the same metaheuristic, while in heterogeneous algorithms different metaheuristics are used.
- *Global/partial* In global hybrids, all the algorithms search in the whole search space. However, the search

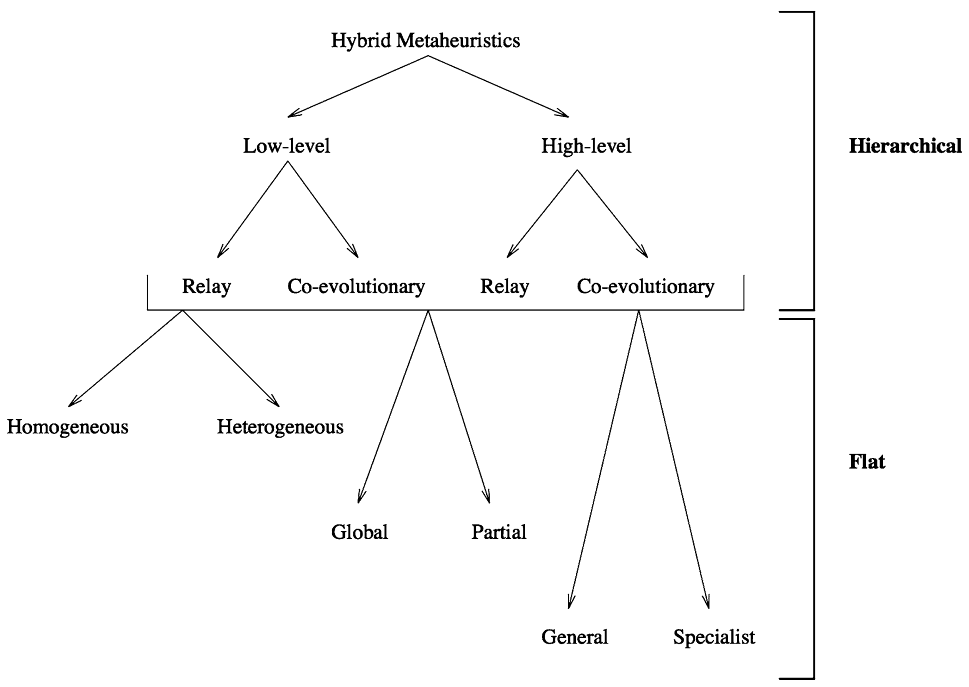


Fig. 3 Classification of hybrid metaheuristics

space is decomposed into subspaces in the partial hybrids.

- *Specialist/general* In a general hybrid, all the algorithms solve the same problem, while specialist hybrids combine algorithms targeted to solve different problems.

Attending to this taxonomy, our model can be classified as a HTH metaheuristic, while several self-contained algorithms cooperate in order to find a solution. Ethane can be classified as well as heterogeneous, global, and general, because two different metaheuristics search in the whole solution space trying to solve the same problem.

3 Description of the model

In this section we present the particularities of Ethane, as well as we briefly outline the proposal of a more generic model being inspired in the chemical compounds called hydrocarbons.

3.1 Ethane

With Ethane we propose a nature inspired heterogeneous parallel search algorithm specifically designed for it execution in heterogeneous hardware platforms.

Usually, using a generic parallel model within a heterogeneous platform leads to bottlenecks caused by islands with limited resources which can not provide good enough solutions to islands whose populations are much more evolved. Furthermore, many communication topologies, as ring, star, torus, cube or hypercube, do not take care of the underlying hardware architecture and could worsen this problem by overloading these slow islands with too much communication. With Ethane, we propose a communication schema where the most of the communication load is distributed over the fastest nodes of the platform, and the slowest ones are placed as their *slaves*.

The chemical compound called ethane consists of two carbon atoms and six hydrogen atoms, joined together with single chemical bonds. We have established a resemblance between the atoms and the computers of the platform, and between the chemical bonds and communication channels. In ethane, each carbon atom is bonded to three hydrogen atoms, and there is another bond between both carbon atoms. In our Ethane algorithm, we propose the same schema, using two basic algorithms resembling different atoms, and migration channels resembling bonds.

For our study, we have implemented two different versions of our algorithm. In Fig. 4 it is shown the schema for the two instances of Ethane studied in this paper. Ethane G (Fig. 4a) assigns a ssGA sub-algorithm to the central carbon nodes, and a SA sub-algorithm to the *slave* nodes. On the contrary, Ethane S (Fig. 4b) allocates a SA sub-algorithm in each one of the central nodes, and a ssGA sub-algorithm in the *slave* nodes. With this schema, most of the communication load falls on the *master* nodes, which are provided with the best hardware, taking some of the load out of the slowest nodes.

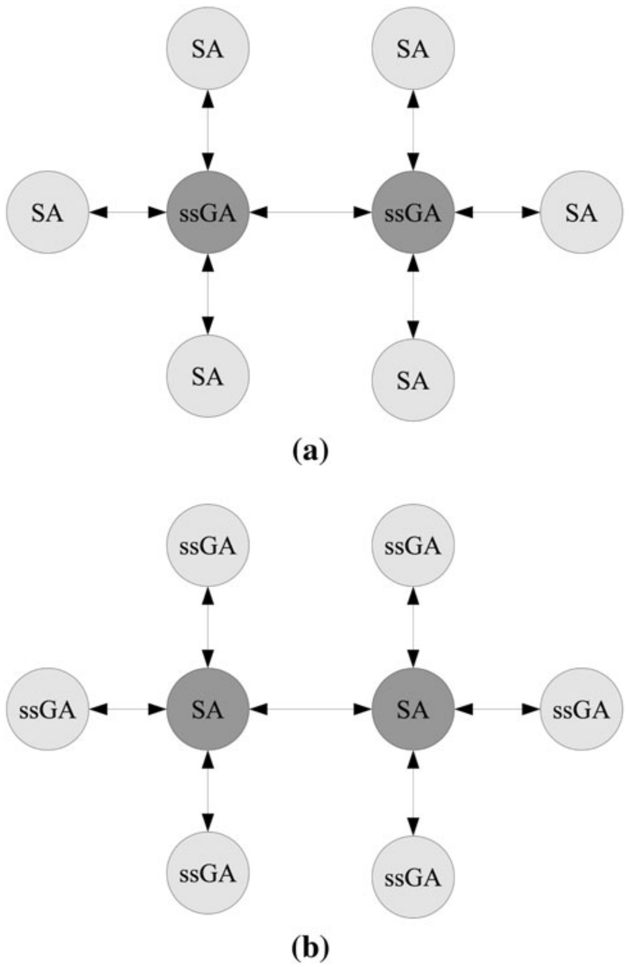


Fig. 4 Communication schema for Ethane G (**a**) and S (**b**)

3.2 An overview of HydroCM

From this chemical inspiration it arises a generic model based on the different structures of hydrocarbons. We have called it HydroCM (HydroCarbon inspired Metaheuristics). We shaped HydroCM as a generic model for a complete family of parallel heterogeneous metaheuristics. The goal of the model is to provide a schema for the islands and communications of the parallel algorithm to efficiently perform a search over heterogeneous hardware architectures.

Figure 5 represents some different structures for hydrocarbons as we can find them in nature. Hydrocarbons are based in only two different atoms, carbon and hydrogen, and each of them can keep a number of bounds, being one for hydrogen and four for carbon.

In our model, we establish a resemblance between computers and atoms in the hydrocarbon. The bonds between atoms have a correspondence to communication channels, and double or triple bonds can be modeled as the amount of information being migrated (intensity of the interaction) or, in the case of non-population based algorithms, a higher

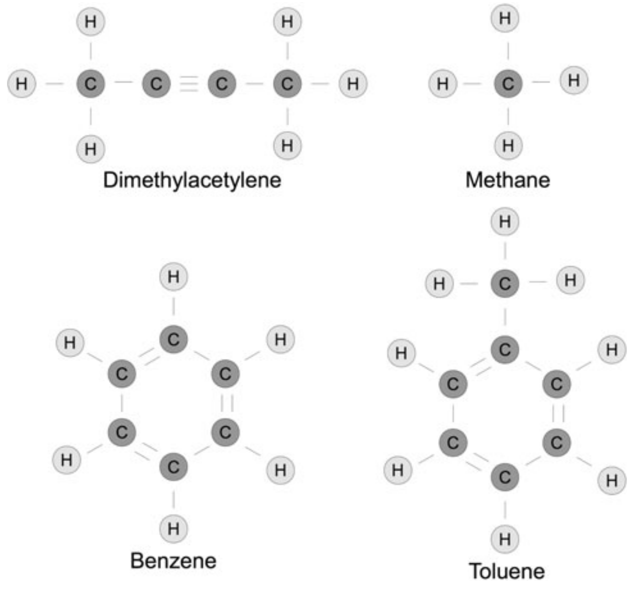


Fig. 5 Different hydrocarbon configurations that can be found in nature; their structures are the basis of HydroCM

migration rate. In our model, the fastest machines are associated with central carbon atoms (because of the higher computational effort caused by the migrations) and the slowest ones are associated with hydrogen atoms.

This model provides us with plenty of different schemes for designing a parallel heterogeneous algorithm because of the amount of hydrocarbons present in nature and their different architectures: linear, ring, branches… obtaining a huge amount of different combinations depending on the number of fast and slow available computers and the topology of the network.

Ethane can be viewed as an instance of HydroCM for an environment composed of eight nodes, where two of them are more powerful than the rest, and making use of ssGA and SA. As well as Ethane is such an instance, we could instantiate many different algorithms depending on the underlying hardware architecture following the model proposed by HydroCM.

4 Performance measures and speedup

In this section we present the performance measures used for assessing the performance of the studied algorithms. The measures that are going to be used are the numerical effort, the total run time, and the speedup. We will also analyze the evolution of the fitness and the evolution of the mean entropy to show the differences in the way each algorithm converge to the optimum.

A widely accepted way of measuring the performance of a parallel metaheuristic is to check the number of evaluations of the fitness function needed to locate an optimum. This performance measure is called *numerical effort*. Numerical effort is widely used in the field of metaheuristics because it removes the effects of the implementation and the platform, but it could be misleading in many cases for parallel methods if used alone. Furthermore, the goal of the parallelism is not the reduction of the number of evaluations (this is a goal for decentralized algorithms) but the reduction of the running time.

The most significant performance measure for a parallel algorithm is the total run time needed to locate a solution. In a non-parallel algorithm, the use of the *CPU time* is a common performance measure. While parallelizing an algorithm should definitely include some overhead, for example for communications, we are not able to use only the *CPU time* as a performance measure. Since the goal of parallelism is to reduce the real time needed to solve the problem, for parallel algorithms it becomes necessary to measure the real run time (wall-clock time) to find a solution.

Because of the non-deterministic behavior of metaheuristics, average values for time and numerical effort are usually needed. We have executed the tests 100 times in this chapter in order to perform a rigorous statistical analysis.

In our analysis we will also study the speedup. The speedup s_m (Eq. 1) represents the ratio between sequential and parallel average execution times ($E[T_1]$ and $E[T_m]$ respectively).

$$s_m = \frac{E[T_1]}{E[T_m]} \tag{1}$$

For the speedup to be a meaningful metric, we have to take care of several aspects for its analysis. Because of the aforementioned non-deterministic behavior of metaheuristics it is necessary to use average times, being these times the wall-clock times. The algorithms run in the single and multiprocessor platform must be exactly the same, thus panmictic algorithms can not be used for the speedup analysis. The algorithms have to be executed until they find the optimum or a solution of the same quality (Alba 2002). Since in our study we are working over a heterogeneous platform, our reference point is the execution time of the program on the fastest single processor.

In order to analyze the convergence speed of the algorithms we will analyze their behavior from two different points of view: the best fitness reached and the diversity of the population (Folino et al. 2003). From the first point of view we will analyze the evolution of the *global best fitness* of each algorithm. From the point of view of the diversity we will analyze the mean population entropy as a way of monitoring the information quantity of the population. The mean entropy H of a population $P(t)$ at a time t is defined in Eq. 2, where $i : P_0^i$ is the proportion of 0's at string position $i : 1, \ldots, l$, and $i : P_1^i$ is the proportion of 1's.

$$H[P(t)] = -\frac{1}{l} \cdot \sum_{l}^{i=1} (P_0^i \cdot \log_2 P_0^i + P_1^i \cdot \log_2 P_1^i) \tag{2}$$

5 Problems, parameters, and platform

In this section we include the basic information necessary to reproduce the experiments that have been carried out for this article. First we will present the set of benchmark problems used for assessing the performance of our proposal. Second we will briefly explain the parameters used within the sub-algorithms, and then the underlying hardware and software platforms.

5.1 Benchmark problems

In order to assess the performance of our algorithms, we have used four problems in the analysis: the subset sum problem (SSP) (Jelasity 1997), the massively multimodal deceptive problem (MMDP) with 6 bits (Goldberg et al. 1992), the multimodal problem generator P-PEAKS (De Jong et al. 1997) and the well-known MAXSAT problem (Garey and Johnson 1990). We have chosen this set of problems because they show some features that are interesting for our study, like complexity, epistasis, multimodality, deceptiveness, and the features of a problem generator.

(1) *SSP* The SSP problem consists in finding a subset of values $V \subseteq W$ from a set of integers $W = \{w_1, w_2, \ldots, w_n\}$, such that the subset sum approaches a constant C without exceeding it. We have chosen an instance with 2,048 random integer numbers in the range $[0 \cdots 10^4]$ following a Gaussian distribution, being the value of the sum for the optimum $C = 3{,}256{,}234$.

    We formulate SSP as a maximization problem whose fitness is given by the function:

$$f(\vec{x}) = a \cdot P(\vec{x}) + (1 - a) \cdot \max[C - 0.1 \cdot P(\vec{x}), 0] \tag{3}$$

where $P(\vec{x})$ is the sum of the integers of a tentative solution, and $a = 1$ when $\vec{x}$ is admissible $(C \geq P(\vec{x}))$ and $a = 0$ otherwise. Solutions exceeding C are penalized.

(2) *MMDP* MMDP is one of so called deceptive problems. Deceptive problems are specifically designed to make the algorithm converge to wrong regions of the search space, decorrelating the relationship between the fitness of a string and its genotype. In MMDP a binary string encodes k 6-bit sub-problems which contribute with a partial fitness depending on its number of 1's (unitation) following Table 1. We have

used an instance with strings of 150 bits so that the global optimum is $k = 25$.

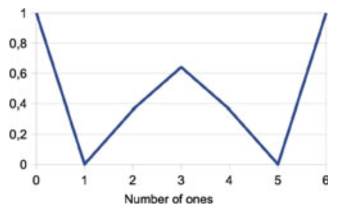
(3) *P-PEAKS* The P-PEAKS problem is a multimodal problem generator. Using a problem generator, we evaluate our algorithms on a high number of random problem instances, since a different instance is solved each time the algorithm is run, removing the opportunity to hand-tune algorithms to a particular instance. The idea behind P-PEAKS is to generate P random N-bit strings that represent the location of P peaks in the search space. The fitness value of a string is the number of bits the string has in common with the nearest peak in the space divided by N (Equation 4). In this work we have used an instance of $P = 1{,}000$ peaks and a length of $N = 2{,}048$ bits.

$$f_{P-PEAKS}(\vec{x}) = \frac{1}{N} \max_{1 \le i \le p} \{N - HammingD(\vec{x}, Peak_i)\} \qquad (4)$$

(4) *MAXSAT* In MAXSAT problem we try to satisfy the maximum number of clauses of a logical expression. Formally this problem is formulated as follows. Being $U = \{u_1, \ldots, u_n\}$ a set of n logical variables. An assignment for u is a function $t : U \rightarrow \{true, false\}$. The literal u (or $\neg u$) is true conditioned by assignation t if and only if $t(u) = true$ (or $t(\neg u) = false$). A clause is defined as a set C of literals that are connected by the disjunction. The set of clauses is called formula. A formula f consists of the conjunction of its clauses (conjunctive normal form). An assignment t satisfies a formula f if and only if t satisfies all the clauses in f. Each clause C is satisfied if there exists, at least, a literal $u \in C$ which is true conditioned by assignation t. The SAT problem lies in, given a formula f, find an assignment t which satisfies these formula or expression. The MAXSAT problem consists in finding an assignment t that maximize the number of satisfied clauses within a formula f. The MAXSAT instance used in our analysis consists of 75 variables and 325 clauses, existing an assignment which makes true all the clauses, thus the global optimum value is 325. This instance has been obtained from SATLIB (Hoos and Sttzle 2000) and is an instance from the phase transition region (Cheeseman et al. 1991) of the Random-3-SAT problem.

Table 1 Bipolar deception (6 bits) sub-function value

#ONES	sub-function value
0	1.000000
1	0.000000
2	0.360384
3	0.640576
4	0.360384
5	0.000000
6	1.000000



5.2 Parameters of the algorithms and platform

The parameters used in every ssGA sub-population are: a population size of 64 individuals, a crossover probability of 0.8 and a mutation probability of 1.0 divided by the chromosome length. The genetic operators are a single point crossover and a bit flip mutation. For the SA, we used the same mutation probability. For the SSP and P-PEAKS problems the chromosome length is 2048, for MAXSAT its length is 75, and in the case of MMDP its length is 150 for both algorithms. In the case of the panmictic ssGA, the population size has been also set to 64 individuals.

Several initial preliminary experiments were run to establish a migration schema, taking care of the CPUs and network speeds, as well as the complexity of the problems. We have chosen a migration frequency of 200 iterations for all the configurations for being the best performing tested rate (among 10, 25, 50, 100, 200 and 500) for the algorithms in the ring topology. The number of individuals migrated is 1 in all cases. For the ssGA, the emigrant is randomly selected and the immigrant always replaces the worst individual of the population. In the SA, the immigrant is treated as a new move. As we noted before, the communication takes place asynchronously.

The hardware infrastructure used in our analysis (Fig. 6) consists of 8 different machines: two of them have an Intel Core 2 Quad Q9400 @ 2.66 GHz processor and 4 GB of RAM (namely Type A, fast), the other six computers have an Intel Pentium 4 @ 2.4 GHz processor and 1 GB of RAM (namely Type B, slow). All the computers are managed by a GNU/Linux distribution, being Debian 5.0 for Type A, and SuSE 8.1, Debian 3.1 and Ubuntu 6.10 for Type B. The computers are connected by a Gigabit Ethernet Network. The algorithms have been implemented in Java in order to support both hardware and software heterogeneity. For the purpose of the analysis the version 1.6.0_01 of the Java Virtual Machine (JVM) is used in all the nodes.

6 Tests and analysis

In this section we analyze the behavior of Ethane, and compare it with the well-known ssGA unidirectional ring, as well as a SA unidirectional ring. As we can see in Table 2, the results of the panmictic versions of ssGA and SA were not competitive with the parallel models (they needed much more time or were not able to find the optimum), thus in the forthcoming analysis we are not going to

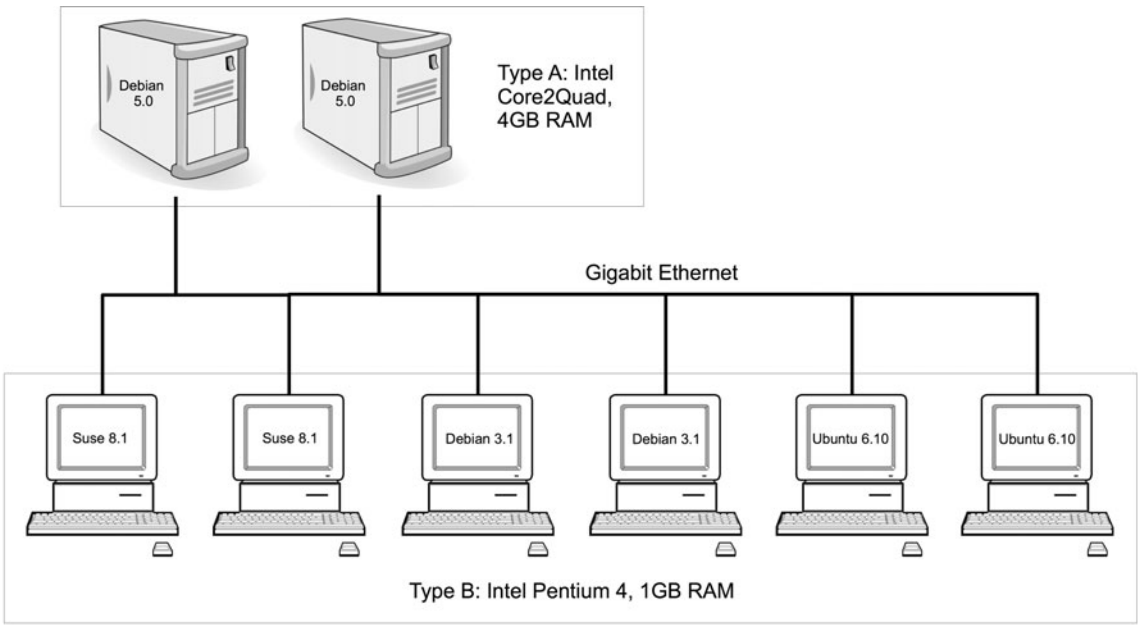


Fig. 6 Schema of the infrastructure showing the heterogeneous hardware and software systems used in our study

study their behavior. We have analyzed the aforementioned performance measures, being numerical effort, total run time and speedup. In the next section we will analyze the evolution of the fitness and the mean entropy.

We have implemented two different algorithms based on Ethane. For the first one, Ethane G, we have provided the Type A computers with a central ssGA island, and Type B computers with a SA island. For the second algorithm, Ethane S, the fastest machines run central SA islands and the slowest ones run ssGA. As we mentioned above, the migration scheme resembles a molecule of ethane as represented in Fig. 4. In the parallel ssGA and SA used as reference, the islands have been distributed over a unidirectional ring, placing the most powerful computers in the first and fourth place in a sort of MaxSumSort (Branke et al. 2004).

All the data obtained during the testing is summarized in Table 2. We have included statistical measures like mean, median, standard deviation, range, skewness, and kurtosis to better understand the results and make a fair comparison. As we do not know the statistical distribution of the data, they have been statistically compared with Mann–Whitney U test, using a value of 95 % for the confidence.

6.1 Numerical effort

In Table 2 we have presented different statistical measures for numerical effort and total run time. In the first place we are going to analyze the numerical effort.

For the SSP, we can see that both Ethane G and Ethane S performed clearly better than ssGA and SA rings, attending to both the mean and the median values for numerical effort. The ssGA ring median number of evaluations was more than four times higher than Ethane G or S, and for the SA ring it was almost 20 times higher. Attending to dispersion and asymmetry measures, we can see how the standard deviation and the range was also better (more robust) for our proposal, and they obtained quite good values for the skewness too. While the value for standard deviation and range is much lower for our proposal, the high value for the kurtosis in Ethane G suggests that the behavior of our proposal described a distribution with higher *peakedness*, denoting a more stable behavior.

In the case of MMDP, the SA ring was not able to achieve the global optimum in a reasonable time, and ssGA ring obtained only a 97 % of hits, while our proposal always reached the optimum. However, ssGA ring was the algorithm with the lower mean and median number of evaluations. Ethane S obtained also good values for the median and the mean, but attending to the diversity in the distribution, both Ethane G and S obtained better values for the standard deviation and range, as well as their distributions showed lower asymmetry and a higher concentration, showing again a more robust behavior.

For P-PEAKS, the mean and median values for numerical effort showed again that the best performing algorithm was Ethane G, being Ethane S the second best algorithm attending to the median value. From the point of view of the shape of the distribution, Ethane G was patently the more stable algorithm, showing the best values for standard deviation and range, and almost a null asymmetry.

In the case of MAXSAT, in the first place we have to talk about the hit percentage; Ethane S and ssGA ring reached the optimum 89 and 92 % of the times, while Ethane G only

Table 2 Numerical effort (number of evaluations) and total run time (ms) for the tested models and panmictic algorithms

Algorithm	Problem	Time (ms)						Numerical effort (evaluations)						%Hit
		Mean	Median	Std. dev.	Range	Skewness	Kurtosis	Mean	Median	Std. dev.	Range	Skewness	Kurtosis	
Ethane G	SSP	**3,839**	**1,742**	**3,969**	**15,510**	1.73	**2.12**	**53,501**	**23,644**	**56,731**	*219,478*	1.73	**2.12**	**100**
	MMDP	**60,500**	**57,704**	**16,112**	**75,596**	0.93	**0.88**	7,894,402	7,519,063	*2,111,716*	*9,865,070*	0.93	**0.86**	**100**
	P-PEAKS	**265,392**	**264,863**	**19,579**	**75,795**	**0.34**	−0.73	**96,601**	**95,859**	**7,181**	**28,417**	**0.35**	−0.66	**100**
	MAXSAT	*1,419*	*1,206*	*861*	**3,315**	*1.23*	*1.26*	**76,796**	*65,624*	*68,606*	*348,898*	2.24	**7.83**	49
Ethane S	SSP	*7,744*	*3,584*	*7,753*	*27,241*	**1.27**	0.39	*59,824*	*26,857*	*60,635*	**214,606**	*1.27*	0.40	**100**
	MMDP	105,721	103,136	*35,158*	*134,859*	**0.49**	*−0.52*	*5,814,110*	*5,590,693*	**1,940,766**	**7,416,717**	**0.51**	*−0.49*	**100**
	P-PEAKS	990,575	770,243	420,473	1,165,600	*0.48*	−1.57	177,915	*139,225*	*75,722*	108,136	0.49	−1.57	**100**
	MAXSAT	**1,327**	**999**	946	*3,774*	1.59	**1.67**	*79,044*	**52,594**	**65,035**	**264,143**	1.64	1.86	*89*
ssGA ring	SSP	10,304	7,292	7,805	33,242	1.60	*1.85*	148,978	103,828	113,894	483,237	1.59	*1.86*	**100**
	MMDP	*87,487*	*68,111*	66,687	240,513	*0.72*	−0.69	**4,554,597**	**3,437,374**	3,465,438	12,153,436	*0.68*	−0.76	97
	P-PEAKS	*273,214*	*272,035*	*20,680*	*90,283*	0.70	**0.21**	*144,615*	144,284	111,137	*47,621*	0.61	**−0.09**	**100**
	MAXSAT	1,461	1,387	**488**	115,777	**0.98**	0.59	117,410	115,295	475,333	12,785,321	**1.02**	0.93	**92**
SA ring	SSP	23,197	17,643	24,448	97,354	*1.28*	1.02	541,101	469,585	504,567	1,875,914	**0.90**	−0.11	**100**
	MMDP	*	*	*	*	*	*	*	*	*	*	*	*	0
	P-PEAKS	655,957	639,242	90,641	392,804	0.77	*0.17*	234,326	229,526	27,438	110,431	*0.43*	*−0.58*	**100**
	MAXSAT	33,835	25,730	43,378	105,727	1.54	**2.51**	9,326,920	7,002,249	12,069,594	29,394,205	*1.54*	*2.50*	5
Std. ssGA	SSP	16,387	11,871	17,004	74,565	1.76	3.94	141,483	102,812	146,972	645,407	1.76	3.95	100
	MMDP	*	*	*	*	*	*	*	*	*	*	*	*	0
	P-PEAKS	1,156,463	1,133,407	105,875	421,895	0.77	0.11	64,972	63,204	5,914	23,855	0.95	0.64	100
	MAXSAT	22,811	5,094	27,554	68,649	0.77	−1.32	772,717	165,576	934,594	2,331,288	0.77	−1.31	18
Std. SA	SSP	16,536	11,994	19,118	98,519	3.21	12.76	159,069	116,375	186,127	958,855	3.23	12.85	100
	MMDP	*	*	*	*	*	*	*	*	*	*	*	*	0
	P-PEAKS	713,858	691,873	145,817	637,032	0.32	0.08	40,399	39,162	8,268	36,131	0.32	0.09	100
	MAXSAT	*	*	*	*	*	*	*	*	*	*	*	*	0

Bold numbers identifies the best value; italic number identifies the second better value obtained a 49 % of hit and SA ring a lower 5 %. Attending to the mean and median values, our algorithms obtained the best results again, clearly outperforming the ssGA and SA rings.

The diversity and shape measures for the distribution were also better for our proposal, showing a much more stable behavior. In summary, attending to the numerical effort, our proposals were the best performing algorithms in 3 out of 4 problems, and were also more robust in all the benchmark problems, obtaining better values for diversity and shape measures. All the differences noted in this subsection are statistically significant according to the Mann–Whitney U.

6.2 Total run time

Now we are going to study a more meaningful metric in a parallel algorithm, the total run time. Table 2 also contains the data for the run time.

For the SSP, as for the numerical effort, we can see that both Ethane G and Ethane S performed clearly better than ssGA and SA rings from the point of view of the wall-clock time. The mean values were quite lower for our proposal, and the median was even more better. From the point of view of dispersion and asymmetry measures, we can see how the standard deviation and the range was also clearly better for our proposal. They obtained quite good values for the skewness too, and the lower deviation and higher kurtosis suggest a more shaped distribution for Ethane G (values are more concentrated), showing a robust behavior.

As we noted in the previous subsection, for the MMDP the SA ring was not able to achieve the global optimum and ssGA ring obtained a 97 % of hits, being a 100 % in both Ethane versions. Although ssGA ring performed a lower number of evaluations, attending to the total run time, Ethane G was the best performing algorithm. Furthermore, its values for diversity and concentration were the best too. Ethane S obtained also good values for the diversity measures, being the ssGA ring the worst algorithm attending to these measures. Our proposal showed again a more stable behavior than the reference models.

As we can see in Table 2, for P-PEAKS problem generator, Ethane G was again the best performing and more stable algorithm, obtaining the best values for mean, median, standard deviation and range, and the lower skewness. In this case the difference was not very high, but statistically significant, so the behavior of the proposed Ethane G was better and more robust, although it was also good and robust for the ssGA ring.

For the case of MAXSAT, we have to remember that only Ethane G and ssGA ring were able to obtain the optimum in almost all the runs. Looking at the mean and median values, our algorithms obtained the best results again, but in this case the difference is not very high comparing to the ssGA ring. However, the dispersion measures for the distribution were much better for our proposal, showing a much more stable behavior.

Attending to the total run time, which is the most meaningful metric in a parallel system, our proposals were the best performing and also more robust in all the benchmark problems, obtaining better values for the mean, median, deviation, and shape measures. Again, all the differences noted in this subsection are statistically significant according to the Mann–Whitney U test with more than a 95 % confidence.

6.3 Speedup

In the Table 3, we can see a summary of the execution time of the studied algorithms within the fastest processor and its speedup with respect to its own execution in the eight processors heterogeneous platform. As we can see, both versions of Ethane have obtained a better speedup than the ssGA and SA rings in 3 out of 4 benchmark problems.

As it is shown in Table 3, Ethane G has performed better than the reference ssGA ring even in a single processor in the case of SSP. Even when its performance over a single processor is still better, its speedup is the best of the four models, however, the value for the speedup is not good for any of the algorithms for this problem, being the value for Ethane G 4.01×.

In the case of the MMDP, Ethane G obtained again the best speedup value, although it was not a high value. The

Table 3 Time (ms) for the tested models in a single processor and its speedup

Algorithm	Subset sum		MMDP6		P-PEAKS		MAXSAT	
	Avg. time	Speedup	Avg. time	Speedup	Avg. time	Speedup	Avg. time	Speedup
Ethane G	**15,385**	**4.01**×	149,495	**2.47**×	6,453,594	24.32×	7,565	**5.33**×
Ethane S	18,905	2.45×	**128,310**	1.21×	5,991,042	6.04×	**3,251**	2.45×
ssGA ring	25,074	2.43×	911,108	1.04×	9,190,826	**33.64**×	3,675	2.51×
SA ring	92,617	3.98×	*	*	**5,278,337**	8.05×	166,998	4.93×

Bold numbers identifies the best values

speedup for Ethane S and ssGA ring was almost null, because of the light computational effort required for this problem.

For the P-PEAKS problem, Ethane G, ssGA ring and SA ring were able to achieve super-lineal speedup, being the best value obtained by ssGA ring. However, Ethane G was faster than ssGA ring when run in a single processor. The values for the speedup achieved an impressive 33.64× for ssGA ring and 24.32× for Ethane G.

In the MAXSAT case, the higher speedup value was obtained by Ethane G, with a value of 5.33×. The speedup was more than twice the one obtained by ssGA ring, while the distributed SA ring obtained a good value because of its poor performance within a single processor.

In summary, the speedup of the proposed and reference algorithms was not really high except for the P-PEAKS problem, but in 3 out of 4 benchmark problems our proposal was able to obtain the best speedup values. This fact could be explained by the huge difference among the power of the different hardware configurations used (remember that the reference point for speedup is the best performing processor, a difficult challenge). We can see that the value for the speedup was higher with the problems where more computational effort was needed.

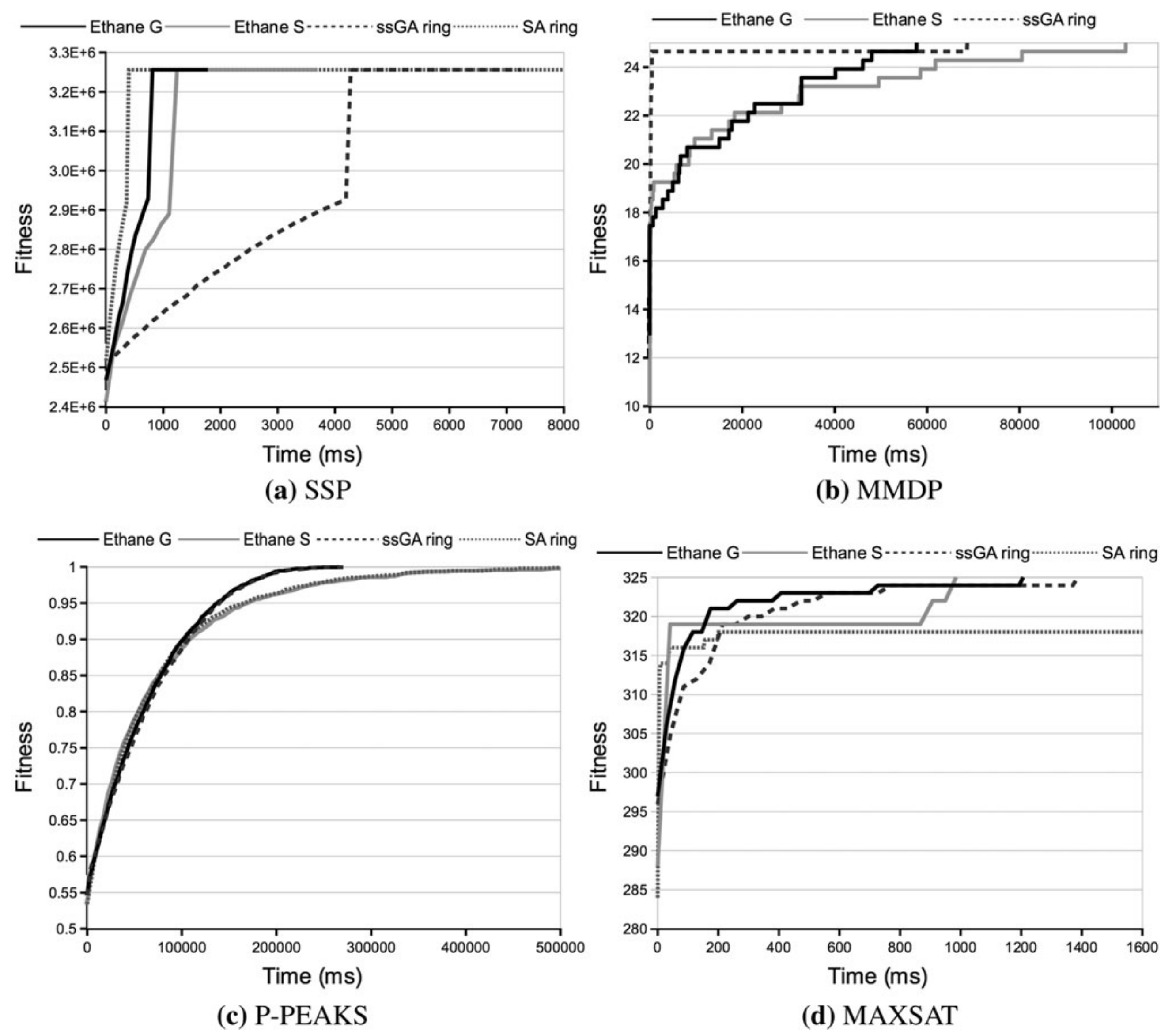


Fig. 7 Evolution of the fitness. **a** SSP, **b** MMDP, **c** P-PEAKS, and **d** MAXSAT

7 Analysis of the diversity

In this section we will graphically analyze the evolution of the fitness and the diversity of the population during the search. For this purpose we are going to track the value of the best fitness found and the mean population entropy along the search.

7.1 Evolution of the fitness

Figure 7a–d is showing the evolution of the best value for the fitness along the median execution of each algorithm for each problem.

In the case of SSP (Fig. 7a), the figure shows that both Ethane versions clearly outperforms the ssGA ring, converging to the optimum quite faster. We can see how Ethane G performs even better than Ethane S for this problem. The SA ring converged even faster than both Ethane versions, but it got stuck in a local optima, finding the global optimum in an almost ten times higher time than the median value for Ethane G. The table is not showing the complete curve for SA ring in order to maintain the representativity for the rest of the algorithms.

For the MMDP (Fig. 7b), Ethane G performed better than Ethane S, obtaining a quasi-lineal convergence curve. The ssGA ring converged faster than Ethane G at the

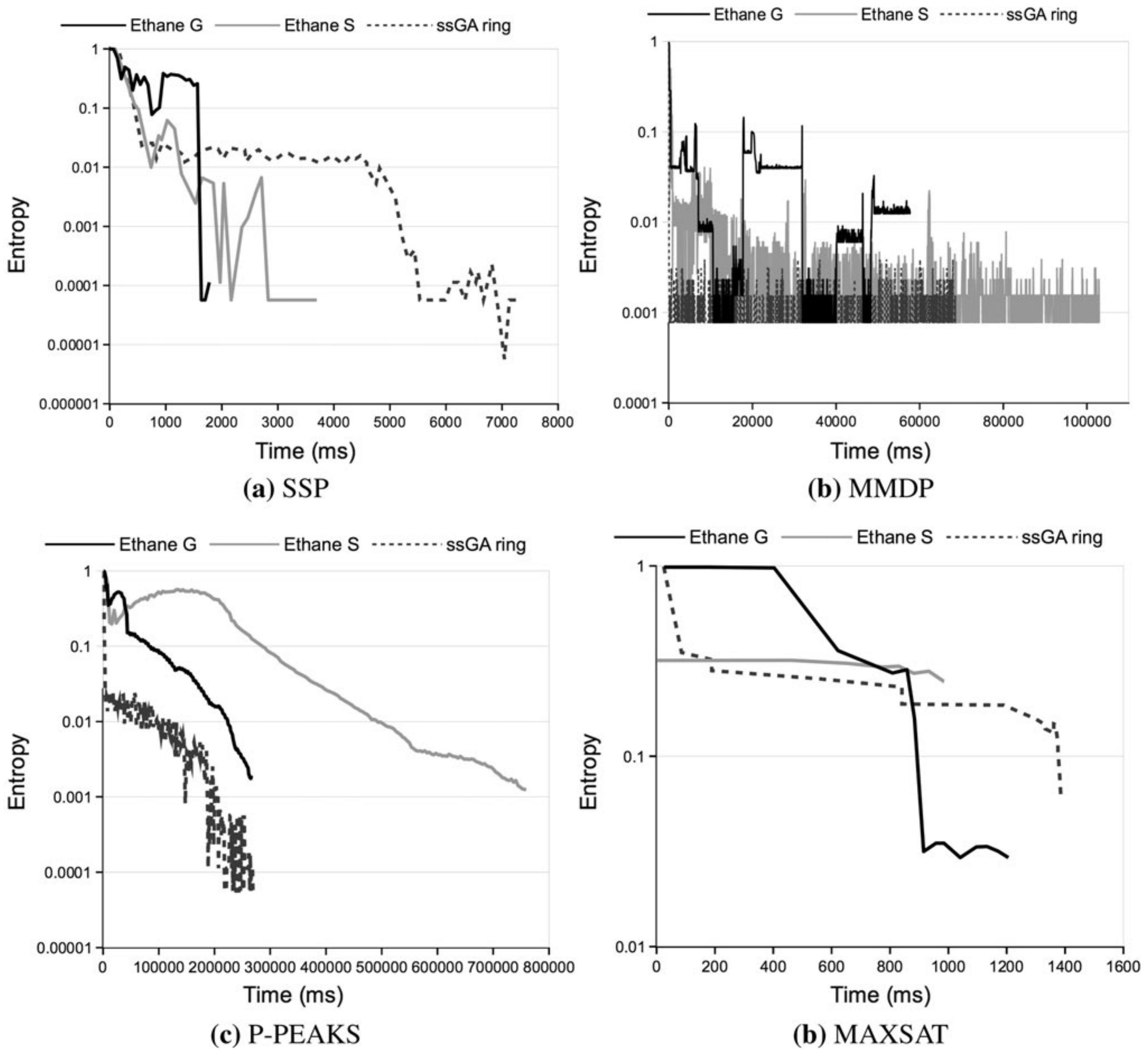


Fig. 8 Evolution of the entropy. **a** SSP, **b** MMDP, **c** P-PEAKS, and **d** MAXSAT

beginning of the run, but its fitness curve was stuck close to the optimum and was not able to find it in a better time than Ethane G. As we will see in the next subsection, this convergence speed was correlated with a diversity drop.

In the case of P-PEAKS (Fig. 7c), we can clearly distinguish between the better performing Ethane G and ssGA ring, and the worse Ethane S and SA ring. However, it is hard to distinguish between Ethane G and ssGA ring, while both of them are showing a good convergence curve. The analysis of the diversity will help us for the distinction of their behavior.

For MAXSAT (Fig. 7d), we can see how both Ethane versions were able to outperform the ssGA ring. The SA ring was not competitive, so the table is not showing its complete convergence curve. We can notice a difference in the behavior of Ethane G and Ethane S; in Ethane G, the best fitness grows constantly in a manner similar to ssGA ring. However, Ethane S was stuck in a local optimum during most of the search time, showing a behavior more similar to the SA ring.

7.2 Analysis of the population entropy

In Fig. 8 we present the evolution of the mean population entropy along the search. We are going to analyze these figures and compare them with the aforementioned evolution of the fitness.

Comparing the Figs. 7a and 8a, we can see how for SSP, although Ethane G and S converged faster than ssGA ring, they were able to maintain a higher diversity within its population during the search. The balance between exploration and exploitation was better in our proposal, achieving a fast convergency while maintaining the population diversity.

For MMDP, if we look at Fig. 8b, we can see that, again, Ethane G and S maintained a higher diversity during the whole search. Comparing with Fig. 7b, we can see how the fitness curve has a maintained grow rate for Ethane. However, for the ssGA ring, the curve is stuck during the whole search while the diversity drops.

In the case of P-PEAKS, looking at the convergence curve does not provide us with much information but, if we compare Figs. 7c and 8c we realize that, while the evolution of the best fitness is very similar between Ethane G and the ssGA ring, our proposal was able to maintain a higher diversity, as well as Ethane S does. Thus, Ethane G obtained again the best balance between exploitation and exploration.

For the MAXSAT problem, the results are similar to the rest of the benchmark problems. In Fig. 8d we can see how both Ethane versions were able to maintain a highest diversity within the population. In this case the ssGA ring was also able to maintain a good diversity during the search.

As a summary, we can say that Ethane G and S have shown a better balance between exploitation and exploration than the reference ssGA ring in all the benchmark problems, as their fitness curve has grown faster than the reference in the majority of the cases maintaining a higher diversity in all the problems.

8 Conclusions and future work

In this paper we have presented a new heterogeneous parallel search algorithm based on the structure of ethane. We have also shaped a general model for designing heterogeneous algorithms depending on the underlying heterogeneous platform, inspired in the structures of the hydrocarbons present in nature.

We have performed a set of tests in order to assess the performance of our proposal, and compared it with two well-known state-of-the-art models, both ssGA and SA unidirectional ring, and two well-known panmictic algorithms: SA and ssGA. Our tests have shown that our proposal can perform better in terms of time and numerical effort than the reference models, and Ethane is even able to find the solutions in a more robust/stable manner. Also the speedup of the proposed models is competitive with that of the reference models, obtaining quite good values even with the huge differences between the performance of the computers of the heterogeneous platform.

We also analyzed the exploitation/exploration balance analyzing the relationship between the fitness evolution speed and the population diversity. We have shown that Ethane can perform a search in a fastest way than the reference models do while maintaining a higher diversity within the population, exposing a better balance between exploitation and exploration.

As future work we propose to extend and deeply analyze the HydroCM model, as well as to assess its performance with different configurations and real-life applications. Our goal is to offer a general model for gracefully matching computers of different powers to run different algorithms to efficiently solve the same problem, in a way that an heterogeneous platform does not constitute a problem but, on the contrary, could be used as a target platform for specialized new parallel algorithms.

Acknowledgments Authors acknowledge funds from the Spanish Ministry MICINN and FEDER under contracts TIN2011-28194 (roadME) and TIN2008-06491-C04-01 (M* http://mstar.lcc.uma.es) and CICE, Junta de Andalucía, under contract P07-TIC-03044 (DIRICOM http://diricom.lcc.uma.es).

References

Aarts EHL, Verhoeven MGA (1997) Genetic local search for the traveling salesman problem. Handbook of evolutionary computation. Institute of Physics Publishing and Oxford University Press, Bristol, pp G9.5:1–7

Alba E (2002) Parallel evolutionary algorithms can achieve superlineal performance. Inf Process Lett 82:7–13

Alba E (2005a) Metaheuristics and parallelism. Parallel metaheuristics: a new class of algorithms. Wiley, New York, pp 79–103

Alba E (2005b) Parallel heterogeneous metaheuristics. Parallel metaheuristics: a new class of algorithms. Wiley, New York, pp 395–422

Alba E, Dorronsoro B (2008) The state of the art in cellular evolutionary algorithms. Cellular genetic algorithms. Springer, New York, pp 21–34

Alba E, Troya JM (2001) Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Futur Gener Comput Syst 17:451–465

Alba E, Nebro AJ, Troya JM (2002) Heterogeneous computing and parallel genetic algorithms. J Parallel Distrib Comput 62:1362–1385

Alba E, Luna F, Nebro AJ, Troya JM (2004) Parallel heterogeneous genetic algorithms for continuous optimization. Parallel Comput 30(5–6):699–719

Branke J, Kamper A, Schmeck H (2004) Distribution of evolutionary algorithms in heterogeneous networks. In: Lecture notes in computer science, vol 3102. Springer, Heidelberg, pp 923–934

Cheeseman P, Kanefsky B, Taylor WM (1991) Where the really hard problems are. In: Proceedings of the international joint conferences on artificial intelligence IJCAI-91, Sydney, pp 331–337

Chen H, Flann NS (1994) Parallel simulated annealing and genetic algorithms: a space of hybrid methods. In: Third conference on parallel problem solving from nature, Jerusalem, pp 428–436

Crainic TG, Toulouse M (2003) Parallel strategies for meta-heuristics. Handbook of metaheuristics. Kluwer, Norwell, pp 474–513

De Falco I, Del Balio R, Tarantino E, Vaccaro R (1994) Improving search by incorporating evolution principles in parallel tabu search. In: International conference on machine learning, New Brunswick, pp 823–828

De Jong K, Potter M, Spears W (1997) Using problem generators to explore the effects of epistasis. In: Proceedings of the seventh international conference on genetic algorithms, San Francisco, pp 338–345

Domínguez J, Alba E (2011) Ethane: a heterogeneous parallel search algorithm for heterogeneous platforms. DECIE'11, doi:arXiv:1105.5900v2

Fleurant C, Ferland JA (1996) Genetic and hybrid algorithms for graph coloring. Ann Oper Res 63:437–461

Folino G, Pizzuti C, Spezzano G, Vanneschi L, Tomassini M (2003) Diversity analysis in cellular and multipopulation genetic programming. IEEE Congr Evol Comput (1):305–311

Garey MR, Johnson DS (1990) Computers and intractability; a guide to the theory of NP-completeness. W. H. Freeman, New York

Goldberg DE, Deb K, Horn J (1992) Massively multimodality, deception and genetic algorithms. Parallel Probl Solving Nat 2:37–46

Hoos HH, Sttzle T (2000) SATLIB: an online resource for research on SAT. In: Gent IP, Maaren Hv, Walsh T (eds) SAT 2000. IOS Press, Amsterdam, pp 283–292

Jelasity M (1997) A wave analysis of the subset sum problem. In: Proceedings of the seventh international conference on genetic algorithms, San Francisco, pp 89–96

Lozano M, Herrera F, Krasnogor N, Molina D (2004) real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12(3):273–302

Mahfoud SW, Goldberg DE (1995) Parallel recombinative simulated annealing: a genetic algorithm. Parallel Comput 21:1–28

Martin OC, Otto SW, Felten EW (1992) Large-step markov chains for the TSP: incorporating local search heuristics. Oper Res Lett 11: 219–224

Salto C, Alba E (2012) Designing heterogeneous distributed GAs by efficient self-adapting the migration period. Appl Intell 36:800–808

Salto C, Alba E, Luna F (2011) Using landscape measures for the online tuning of heterogeneous distributed GAs. In: Proceedings of the GECCO'11, Dublin, pp 691–694

Syswerda G (1991) A study of reproduction in generational and steady-state genetic algorithms. Foundations of genetic algorithms. Morgan Kauffman, San Mateo, pp 94–101

Talbi E-G (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8(5):541–564

Talbi E-G, Muntean T, Samarandache I (1994) Hybridation des algorithmes génétiques aveq la recherche tabou. In: Evolution Artificielle (EA94), Toulouse

Voigt H-M, Born J, Santibanez-Koref I (1990) Modeling and simulation of distributed evolutionary search processes for function optimization. In: PPPSN. Lecture notes in computer science, vol 496. Springer, Heidelberg, pp 373–380

Yao X (1995) A new simulated annealing algorithm. Int J Comput Math 56:161–168