An Efficient Probabilistic Population-Based Descent for the Median Genome Problem

Adrien Goëffon
INRIA Bordeaux Sud-Ouest
351 cours de la Libération
33400 Talence, France
goeffon@labri.fr

Macha Nikolski
CNRS / LaBRI
Université Bordeaux I
351 cours de la Libération
33400 Talence, France
macha@labri.fr

David J. Sherman
INRIA Bordeaux Sud-Ouest
351 cours de la Libération
33400 Talence, France
david@labri.fr

ABSTRACT

We present a novel population-based local search algorithm for the *median genome problem*. The primary result of this article is that this probabilistic approach significantly improves the performance of ancestral genome reconstruction compared to existing methods, making it possible to tackle problems where the contemporary genomes may contain many hundreds of markers. Moreover, our method is not limited to triples of genomes, and thus solves the median genome problem in its generality. We show that in real application cases the computational results are highly robust, suggesting that we can interpret the computed median genomes as candidates carrying the semantics of ancestral architectures.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; J.3 [**Life and Medical Sciences**]: *Biology and genetics*

General Terms: Algorithms.

Keywords

Median Genome Problem, probabilistic neighborhood, local search.

1. INTRODUCTION

The increasing availability of fully sequenced genomes has fuelled efforts in understanding the history and function of genomes through comparison of related species and analyses in computational biology. Identifying ancestral genomes architectures is one important question that can now be ad-

[2] implement partial solutions of MGP. Indeed, MGR solves the problem for triples of multichromosomal genomes, while rEvoluzer can treat more than 3 genomes, but only in the unichromosomal case. Both of the proposed solutions are heuristics based on the detection of "good" reversals, operations that are guaranteed to improve the solution. The genomes $\{\Pi_i\}$ are re-written step by step by applying reversals until two (or three) of them become equal. There are two differences in the proposed solutions. First, the definition of what constitutes a good reversal is not exactly the same. Second, when no good reversals remain, MGR performs a k-depth search to find a best reversal, while rEvoluzer allows for backtracking.

In this paper, we present FAUCILS, a new approximate algorithm for MGP in the general case, that is, for an unrestricted number of multichromosomal genomes, while improving performances of existing approaches on restricted instances. The mainly originality of our approach is the definition of a probabilistic neighborhood which evolve within a population-based local search according to observations made on the population. This mechanism allows us to greatly accelerate the search and ensures more convergence, especially for real or structured instances.

2. THE MEDIAN GENOME PROBLEM

A *chromosome* $\pi = (\pi_1, \ldots, \pi_m)$ is represented by a sequence of signed gene markers whose sign indicates their relative direction on the chromosome. A size-n *multichromosomal genome* Π is defined as a set of chromosomes $\{\pi^1, \ldots, \pi^N\}$ such that $\sum_i |\pi^i| = n$. Markers take their value from the set of ordinals $1, \ldots, n$; no given marker appears in more than one chromosome.

For example, $\{(5, -8), (1, 2, -10, 6, 4), (9), (-3, 7)\}$ is a genome of size 10. In [3], concatenation of all chromosomes is represented as a signed permutation.

Given a set of size-n genomes $\{\Pi_i\}$ and a genome distance function d, an instance of the combinatorial minimization problem MGP is defined by two elements $\langle \tau_n, \phi \rangle$:

1. a search space, τ_n, composed of the set of all possible size-n genomes, and

2. an objective function $\phi : \tau_n \rightarrow \mathbb{N}$ (*score*) defined by $\phi(\Pi) = \sum_{\Pi_i} d(\Pi, \Pi_i)$.

A *median genome* for a given set of genomes $\{\Pi_i\}$ is a genome Π that minimizes $\phi(\Pi)$. Every optimal solution to MGP is a median genome.

3. AN ORIGINAL POPULATION-BASED LOCAL SEARCH FOR MGP

For addressing NP-complete problems like MGP in the general case and reaching acceptable solves

48.966356(e)2.1125025(4(0)4990250(2)2)2m(d)-486.302(l)-1.81178(o)-489.86(l)-343.529(o)5.3163715(o)5.3160a)5.31608(t)-416.695(m)-0.060hl chterchneve7

algldr

search processes and crossovers between elements (*individuals*) of a set or multiset of current configurations (*population*) provide intensification and diversification phases.

A local search process applied to many independant replications is sometimes called a *population-based local search* even though there is no interaction between individuals [13]. Here we do not use any crossover operations, but simulate an alternative evolutionary process in order to accelerate the searches and to make multi-start descents more convergent.

We introduce a probabilistic population-based local search algorithm which favours, at each step of the search, the selection of most pertinent neighbors [7] with respect to the population. Structural information about each individual is used to estimate a selection probability at each step of the search. In this process, all replications are dependant, while the descents are carried out simultaneously.

In this section we present a multi-start descent for MGP. We use the descent mechanism presented in section 3.1, adding to each neighbor a selection probability.

Let $\mathcal{P}$ be the population of our population-based descent, which initially contains individuals taken from $\{\Pi_i\}$. Now let us consider a probabilistic function $p : \tau_n \times \tau_n \times \tau_n^{|\mathcal{P}|-1} \rightarrow [0, 1]$, such that $p(\Pi, \Pi', \mathcal{P} \setminus \{\Pi\})$ gives a selection probability of $\Pi' \in \mathcal{R}^1(\Pi)$. Such a probabilistic function is quite similar to the one used for simulated annealing move strategy. The difference here is that only better or equivalent neighbors are accepted by the move strategy, whereas neighbors are generated by a probability distribution (*probabilistic neighborhood* [11]). The aim is not to escape to local optima, but to favour neighbors which share properties with other individuals in $\mathcal{P}$.

This probabilistic function is connected to the notion of *adjacencies*, that we define in the way analogous to Nadeau and Taylor [12].

DEFINITION 1. *Two consecutive elements π_i and π_{i+1} of a chromosome $\pi \in \Pi$ are said to be* adjacent *in* Π. *We note this adjacency by* $(\pi_i.\pi_{i+1})$.

We consider additional adjacencies at the extremities of each chromosome by introducing marker 0. For a chromosome $(\pi_1, \ldots, \pi_m)$, two adjacencies are added: $(0.\pi_1)$ and $(\pi_m.0)$. Notice that $(\pi_i.\pi_j) = (-\pi_j.-\pi_i)$ and $(0.\pi_i) = (-\pi_i.0)$. Finally, we note $\mathcal{A}(\Pi)$ the set of all adjacencies in Π. We have $|\mathcal{A}(\Pi)| = n + N$ (n is the number of markers and N the number of chromosomes).

Each move (rearrangement) breaks one (fission) or two (reversal, fusion, reciprocal translocation) adjacencies. The probabilistic neighborhood encourages adjacencies which are not, or are less, represented in the population to be broken. The probabilistic neighbor selection operates as follows: let $\Pi' \in \mathcal{R}^1(\Pi)$; if $\phi(\Pi') \leqslant \phi(\Pi)$, then Π' replaces Π in $\mathcal{P}$ in function of the proportional representation of the broken adjacencies in $\mathcal{P} \setminus \{\Pi\}$:

$$p(\Pi, \Pi', \mathcal{P}\setminus\{\Pi\}) = 1 - \frac{|\{\Pi'' \in \mathcal{P} \setminus \{\Pi\}, (\mathcal{A}(\Pi) \setminus \mathcal{A}(\Pi')) \cap \mathcal{A}(\Pi'') \neq \emptyset\}|}{|\mathcal{P}| - 1}$$

Algorithm 1 provides an overall view of our probabilistic population-based descent we called FAUCILS for *Fast Ancestor (inference) Using Convergent and Intelligent Local Search*.

Algorithm 1

Require: $\{\Pi_1, \ldots \Pi_k\}$: a set of k multichromosomal genomes of size n; $l.k$: the size of the population; *nbit*: the number of descent iterations.
Ensure: an approximate median genome set $\hat{\mathcal{P}}$
  let $\mathcal{P}$ be the multiset of the current genomes population, which initially contains l copies of each Π_i.
  let $numit \leftarrow 0$ be the number of performed local search iterations
  while $numit < nbit$ **do**
    for all $\Pi \in \mathcal{P}$ **do**
      loop
        randomly select $\Pi' \in \mathcal{R}^1(\Pi)$
        break with probability $p(\Pi, \Pi', \mathcal{P} \setminus \{\Pi\})$
      end loop
      if $\phi(\Pi') \leqslant \phi(\Pi)$ **then**
        $\Pi \leftarrow \Pi'$
      end if
    end for
    $numit \leftarrow numit + 1$
  end while
  return $\hat{\mathcal{P}} = \text{argmin}_{\Pi \in \mathcal{P}}\, \phi(\Pi)$

4. EXPERIMENTS

For experiments we use different kinds of instances: real and random ones, with different numbers of genes and chromosomes by genome.

4.1 Real instances

First we assess our algorithm FAUCILS on two sets of 10 triplets of yeast genomes. The data, provided by Génolevures Consortium[1] (GDR CNRS 2354), consists in five sequenced yeasts from the Kluyveromyces clade: *Kluyveromyces lactis* (Klla), *Saccharomyces kluyveri* (Sakl), *Zygosaccharomyces rouxii* (Zyro), *Ashbya gossypii* (Ergo) and *Kluyveromyces thermotolerans* (Klth). From these data, two sets of permutations have been computed: the first one with 135 markers (K135), and the second one with 499 markers (K499). For the comparison with MGR, which calculates only 3-genomes medians ($N = 3$), we separate in ten instances each possible triplet of genomes: Klla-Sakl-Zyro is K135-1 and K499-1, Klla-Sakl-Ergo is K135-2 and K499-2, ... These five genomes have respectively 6, 8, 7, 6 and 8 chromosomes. We add a real test instance composed by the genomes of Human, Cat and Mouse, and available on the MGR web page[2].

Table 1 shows performances of FAUCILS and MGR on these real instances. FAUCILS is a stochastic algorithm, and two executions may return different results; for each instance we perform multiple executions. Table 1 indicates the best results ϕ_b of 20 executions, their frequency ϱ, the mean scores ϕ_m, the worst scores ϕ_w, the standard deviations σ and the mean computation times of one execution. FAUCILS was run with its default parameters: $l = 3$ (*i.e.* a population size of 9 when $k = 3$), and one million LS iterations (*nbit*); MGR was first run with its default parameters, and secondly with the heuristic option H1 (MGR-H1) for speeding up the search. Each execution was performed on

[1] http://www.genolevures.org
[2] http://nbcr.sdsc.edu/GRIMM/mgr.cgi

Instance	k	n	FAUCILS						MGR		MGR-H1		Δ
			ϕ_b	∱	ϕ_m	ϕ_w	σ	CPU	ϕ	CPU	ϕ	CPU	
K135	5	135	**281**	7/20	281.7	282	0.5	42m	-	-	-	-	-
K135-1	3	135	**168**	1/20	170.6	172	1.1	15m	177	355m	178	44m	-9
K135-2	3	135	**115**	5/20	116.1	117	0.8	14m	119	188m	120	6m	-4
K135-3	3	135	**150**	1/20	151.9	153	0.7	15m	157	348m	160	60m	-7
K135-4	3	135	**132**	14/20	132.3	133	0.5	14m	135	377m	136	13m	-3
K135-5	3	135	**166**	1/20	168.0	169	0.9	15m	173	400m	175	48m	-7
K135-6	3	135	**110**	5/20	111.1	112	0.8	15m	112	148m	114	8m	-2
K135-7	3	135	**160**	1/20	161.7	164	0.9	13m	162	300m	172	26m	-2
K135-8	3	135	**185**	4/20	186.6	188	1.1	13m	193	527m	194	197m	-8
K135-9	3	135	**145**	3/20	146.6	148	1.0	13m	154	296m	154	45m	-9
K135-10	3	135	**159**	2/20	160.6	163	0.9	15m	167	355m	165	30m	-6
K499	5	499	**564**	6/20	565.9	568	1.7	98m	-	-	-	-	-
K499-1	3	499	**407**	5/20	408.1	410	0.9	44m	/	max	413	457m	-6
K499-2	3	499	234	1/20	235.0	236	0.3	38m	/	max	**233**	105m	+1
K499-3	3	499	**338**	6/20	339.1	341	0.9	43m	/	max	339	297m	-1
K499-4	3	499	263	4/20	263.9	265	0.6	39m	/	max	**262**	106m	+1
K499-5	3	499	**372**	2/20	373.7	375	1.0	43m	/	max	375	391m	-3
K499-6	3	499	181	10/20	181.6	183	0.6	38m	**179**	2 days	**179**	74m	+2
K499-7	3	499	**375**	1/20	377.1	379	1.0	37m	/	max	381	317m	-6
K499-8	3	499	**476**	3/20	479.2	481	1.6	40m	/	max	484	823m	-5
K499-9	3	499	**307**	2/20	310.3	310	0.9	35m	/	max	309	230m	-2
K499-10	3	499	**338**	3/20	340.0	343	1.4	42m	/	max	**338**	251m	=
HCM	3	114	**48**	20/20	48.0	48	0	<1m	**48**	10m	**48**	<1m	0

Table 1: Comparison between FAUCILS and MGR on real instances.ϕ_b is the best score returned by FAUCILS, ∱ is its frequency, ϕ_m is the mean score, ϕ_w is the worst score, σ is the standard deviation based on 20 different executions.

a node of Grid'5000[3] and the computational time limit per compute node was fixed to one week. In all tables, Δ gives the difference between the best score returned by MGR and the best score returned by FAUCILS.

From Table 1 one observes that FAUCILS computes better median genomes than MGR. For all the K135 instances, FAUCILS performs better than MGR with 5.9 rearrangements less per instance for the bests runs, and 4.4 rearrangements less in the mean for all the 200 runs (10×20) with low computation times (about 15 minutes against hours for MGR). The MGR H1 heuristic does speed up the program, but the returned solutions are less competitive (except for K135-10 where MGR-H1 beats MGR with default settings). When the number of markers is big (499), MGR needs to be used with its speed resolution heuristic to complete the search, and for these instances FAUCILS and MGR are more comparable in term of scores, although FAUCILS remains better in mean and faster.

Finally, FAUCILS is also robust with more than three genomes (instances K135 and K499) since the returned solutions have very close scores. The increase of the computation time mainly depends on the population size parameter, which can be reduced.

4.2 Random instances

In order to assess the performance of our population-based local search algorithm with respect to the structure and the size of the instance, we generate two types of random instances.

First, we use completely random instances (R) containing a specified number of markers, and a minimum and maximum number of chromosomes by genome (N). On these instances of size 50 and 100, FAUCILS obtains better results than MGR systematically (see Table 2). For larger instances, only one MGR run ended, with an uncompetitive result ($\Delta = -22$). These instances seem to be difficult because of their structure: each genome is a random point of τ_n, and the MGR algorithm seems very dependent on the structure of each instance (see the divergences between all computational times on tables 1, 2 and 3).

In order to estimate the impact of the structure of the instance, we generate simulated instances (S), for which distances between genomes are bounded. An arbitrary ancestral genome is generated from which a specified number of random rearrangements are applied to give three genomes. We specify the number of genes (n) and chromosomes (N), and the number of rearrangements done during the simulation (r); this parameter is an upper bound of the optimal median genome score.

The results are given in table 3. We can see that, with $r = 10$ or $r = 50$, instances are very easy to solve. But when the distances between genomes increase ($r = 100$ and $r = 200$), FAUCILS is very competitive and can find in short computation time solutions considerably better than MGR. Moreover, the algorithm is robust as small values of σ show. For these instances (S), we have to reduce the number of local search iterations to $2000.r$ for an equivalent efficiency.

The evolution of the ratio ϕ/r gives an empirical indication of the structure of the search space. Indeed, for $r = 200$, the minimal number of rearrangements required for reconstructing an evolutionary scenario is about 25% lower than the number of rearrangements made during the simulation. Adding to the relative difficulty to find near-optimal genomes for these instances, we can presume that this ratio represents the quantity of lost information and can be a good indicator for comparing the difficulty of simulated instances.

Finally, we have executed rEvoluzer [2] on each unichromosomal instance: S100-10-1, S100-50-1, S100-100-1, S100-200-1, R50-1, R100-1, R200-1, R-500-1. Except for the three first instances, where rEvolzer founds in few seconds or minutes the sames scores as FAUCILS (10, 50, 95), the program did not return any solution for the five other instances, even given one week of computation.

[3]https://www.grid5000.fr

Instance	n	N	FAUCILS						MGR		MGR-H1		Δ
			ϕ_b	f	ϕ_m	ϕ_w	σ	CPU	ϕ	CPU	ϕ	CPU	
R50-1	50	1	**79**	2/20	80.6	82	0.8	4m	82	6m	83	2m	-3
R50-2	50	1-9	**80**	1/20	81.5	82	0.6	6m	87	5m	87	2m	-7
R50-3	50	3-7	**80**	6/20	80.8	82	0.6	6m	84	7m	85	1m	-4
R50-4	50	5-5	**79**	5/20	80.2	81	0.8	6m	83	8m	84	1m	-4
R100-1	100	1	**171**	1/20	173	175	1.0	8m	175	801m	177	126m	-4
R100-2	100	2-10	**166**	1/20	168.8	170	1.1	10m	176	250m	178	226m	-10
R100-3	100	3-7	**170**	2/20	171.9	174	1.2	10m	174	310m	179	139m	-4
R100-4	100	5-5	**166**	1/20	169.8	172	1.5	10m	169	464m	171	170m	-3
R200-1	200	1	**354**	1/20	357.2	362	1.9	16m	/	max	/	max	-
R200-3	200	10	**351**	1/20	356.1	360	2.0	19m	/	max	/	max	-
R200-4	200	11-20	**344**	2/20	347	350	1.7	21m	/	max	366	4 days	-22
R500-1	500	1	**924**	1/20	928.2	932	2.1	37m	/	max	/	max	-
R500-2	500	10	**942**	2/20	945.4	950	2.3	42m	/	max	/	max	-
R500-2	500	18-44	**936**	1/20	942.7	947	3.1	40m	/	max	/	max	-

Table 2: Comparison between FAUCILS and MGR on random instances

Instance	n	N	div	FAUCILS						MGR		MGR-H1		Δ
				ϕ_b	f	ϕ_m	ϕ_w	σ	CPU	ϕ	CPU	ϕ	CPU	
S100-10-1	100	1	10	**10**	20/20	10.0	10	0	<1m	**10**	<1m	**10**	<1m	=
S100-10-2	100	5	10	**10**	20/20	10.0	10	0	<1m	**10**	<1m	**10**	<1m	=
S100-10-3	100	10	10	**10**	20/20	10.0	10	0	<1m	**10**	<1m	**10**	<1m	=
S100-50-1	100	1	50	**50**	20/20	50.0	50	0	<1m	**50**	8m	51	<1m	=
S100-50-2	100	5	50	**49**	20/20	49.0	49	0	<1m	**49**	8m	**49**	<1m	=
S100-50-3	100	10	50	**49**	20/20	49.0	49	0	<1m	**49**	13m	**49**	1m	=
S100-100-1	100	1	100	**95**	7/20	95.7	97	0.6	<1m	97	190m	98	2m	-2
S100-100-2	100	5	100	**95**	2/20	96.4	98	0.7	<1m	96	55m	98	2m	-1
S100-100-3	100	10	100	**96**	4/20	96.9	98	0.5	<1m	99	70m	99	4m	-3
S100-200-1	100	1	200	**155**	1/20	158.6	160	1.4	5m	163	978m	166	65m	-8
S100-200-2	100	5	200	**145**	1/20	146.6	148	0.7	5m	151	331m	151	67m	-6
S100-200-3	100	10	200	**143**	1/20	145.7	154	2.2	5m	150	114m	154	78m	-7

Table 3: Comparison between FAUCILS and MGR on simulated instances

4.3 Influence of the probabilistic neighborhood

One of the main originalities of FAUCILS is that neighbors are selected with a non-uniform probability. The foremost aim is to select more pertinent neighbors as a function of the similarities between individuals in the current population. Since the population is initialized by the given genomes (instance), the probabilistic selection will have a larger impact on structured instances, that is when genomes share adjacencies; it is notably the case of real data instances.

Figure 2 shows the evolution of the minimum score (in the left) and the average score of all individuals (in the right) during the search for the complete Kluyveromyces genomes instances (K135 at the top, K499 at the bottom). We compare the probabilistic descent (algorithm 1) to the same search without a probabilistic selection (in this case each $p(\Pi, \Pi', \mathcal{P} \setminus \{\Pi\}) = 1$, the search is a simple multi-start FI+SW descent). On both instances the probabilistic selection, which one can view as a dynamic neighborhood reduction, allows the population to converge very quickly without efficiency loss. While the classic search does not stabilize even after one million iterations, the probabilistic one reaches a point of stagnancy after about 200 000 LS iterations. The efficiency is more significant with the largest instance (499 markers). Indeed, the neighborhood is very large in this case, and such a reduction is more effective. Moreover, the augmentation of the number of markers for the same genomes significantly increases the shared adjacencies and consequently the number of non-selected neighbors.

In Figure 3, one observes a similar convergence on a simulated multi-genome instance: 10 genomes obtained after 500 random rearrangements from an initial genome of size 100. On this easier instance, less than 10 000 LS iterations (a few seconds of computation) are necessary to converge to supposed optimal solutions.

On the contrary, on completely random instances (figure 4), both mechanisms have the same efficiency. Indeed, such instances have insignificant numbers of shared adjacencies, and the probabilistic selection has no effect. This random instance has the same size than the simulated one (used in figure 3). We show only the average scores evolution, because for this instance the minimum scores can differ from execution to execution independently from the mechanism used.

This study shows that this probabilistic population-based local search adds semantics to the search in order to reduce the neighborhood. It exploits the structure of the instance for a quick convergence of the population.

5. CONCLUSION

In this paper we proposed a new efficient algorithm for the resolution of the Median Genome Problem (MGP) in the general case. We have notably introduced an novel way for speeding up and making more convergent multi-start descents for the resolution of MGP, especially for real structured instances. The key idea is to use a probabilistic neighborhood which evolves during the search according to the partial results of all descents performed simultaneously.

Experiments realized both on real and random instances show that our software FAUCILS is able to find largely better solutions than MGR, the current reference in the domain. Moreover, this local search approach is very fast and scalable: contrary to other existing techniques, FAUCILS can

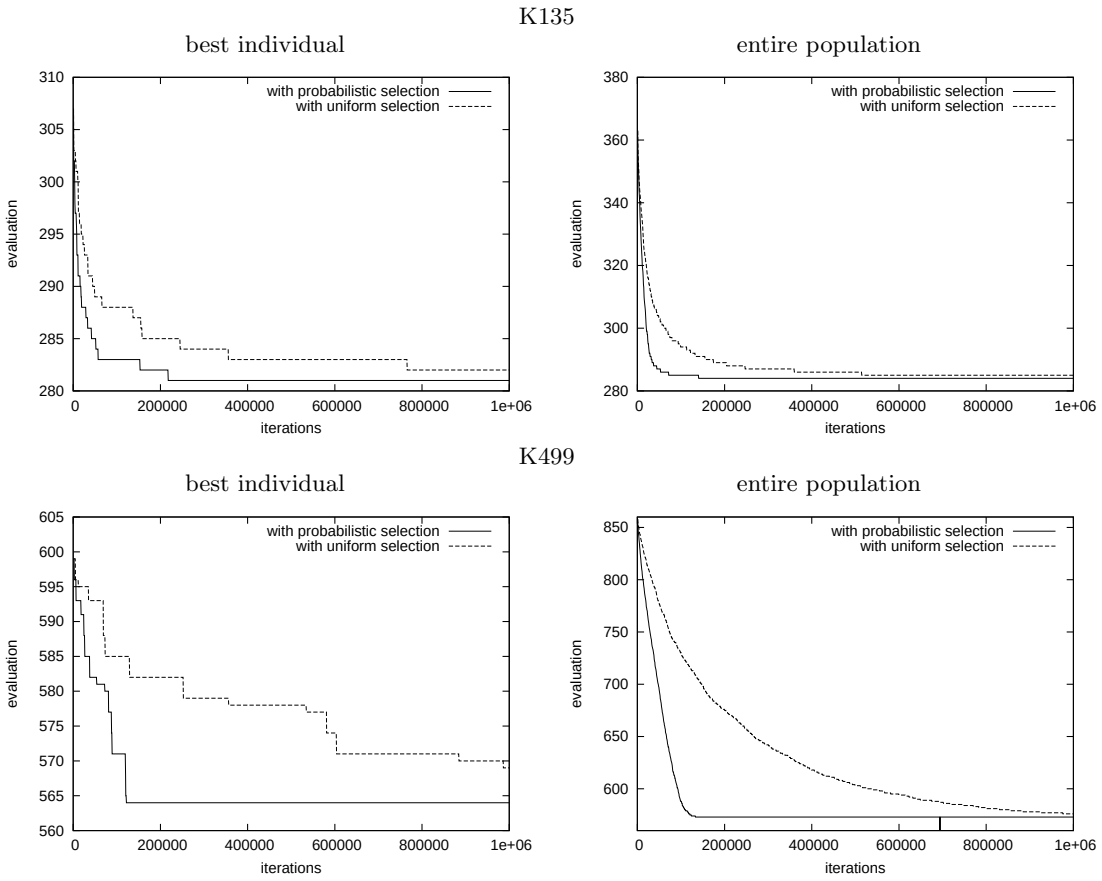

Figure 2: Influence of the probabilistic neighbor selection on real instances: evolution of the minimum and average population score during the search

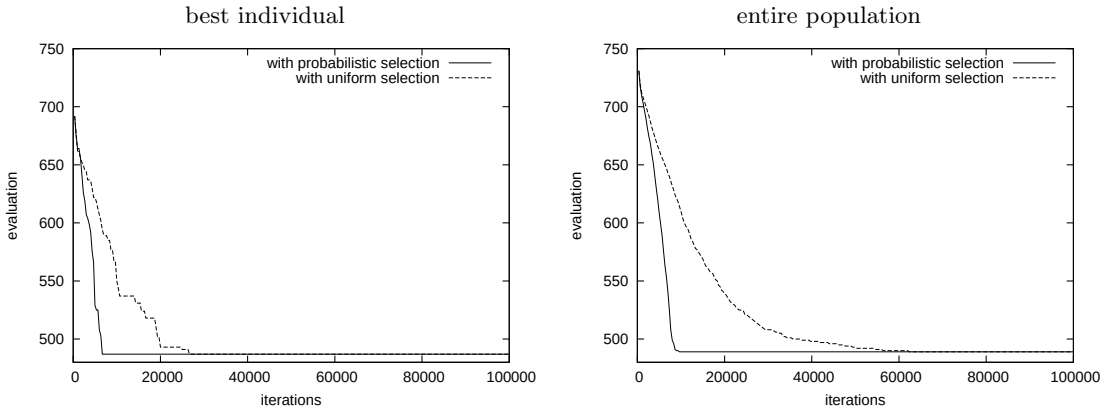

Figure 3: Influence of the probabilistic neighbor selection on simulated instances

treat an unbounded number of multichromosomal genomes, which may contain hundreds or thousands of markers. Future work will involve finding ways to evaluate the quality of solutions in the case of big instances, and to extend this MGP algorithm for the resolution of MGRP. The Median Genome Rearrangement Problem is a very hard computational problem for the resolution of which existing algorithms calculate multiple median genomes.

6. ACKNOWLEDGMENTS

We would like to thank Guillaume Bourque and Matthias Bernt for their helpful comments and for providing us their software MGR and rEvoluzer. We thank Géraldine Jean for the construction of the real signed permutations (Kluyveromyces genomes), and the Genolevures Consortium for making these data available.

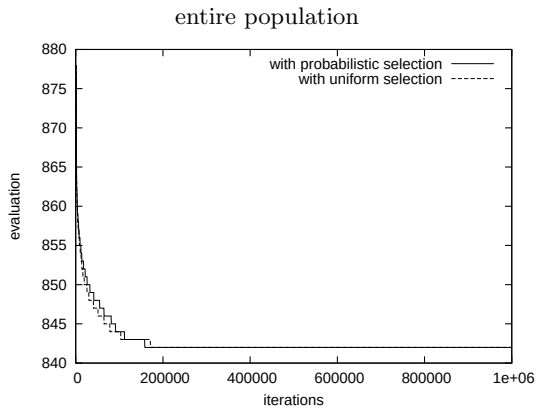

Figure 4: Probabilistic neighborhood has no effet on totally random instances

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see https://www.grid5000.fr).

7. REFERENCES

[1] D. A. Bader, B. Moret, and M. Yan. A linear-time algorithm for computing inversion distances between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[2] M. Bernt, D. Merkle, and M. Middendorf. Genome rearrangement based on reversals that preserve conserved intervals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(3):275–288, 2006.

[3] G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Research*, 12:26–36, 2002.

[4] D. Bryant. The complexity of the breakpoint median problem. Technical Report CRM2579, Centre de Recherches Mathematiques, Universite de Montreal, 1998.

[5] A. Caprara. Formulations and complexity of multiple sorting by reversals. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 84–93, Lyon, France, 1999. ACM Press.

[6] A. Caprara. The Reversal Median Problem. *INFORMS Journal on Computing*, 15(1):93–113, 2003.

[7] A. Goëffon, J.-M. Richer, and J.-K. Hao. Progressive tree neighborhood applied to the maximum parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(1):136–145, 2008.

[8] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pages 178–189. ACM Press, 1995.

[9] H. H. Hoos and T. Stützle. *Stochastic Local Search : Foundations & Applications (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, 2004.

[10] J. Meidanis and J. C. Setubal. *Introduction to Computational Molecular Biology*. PWS Publishing, 1997.

[11] H. Muhlenbein and J. Zimmermann. Size of neighborhood more important than temperature for stochastic local search. In C. Cotta and J. I. van Hemert, editors, *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 1017–1024, 2000.

[12] J. Nadeau and B. Taylor. Lengths of Chromosomal Segments Conserved since Divergence of Man and Mouse. *Proceedings of the National Academy of Sciences of the United States of America, Part 1: Biological Sciences*, 81(3):814–818, 1984.

[13] J. M. Pasia, K. F. Doerner, R. F. Hartl, and M. Reimann. A population-based local search for solving a bi-objective vehicle routing problem. In C. Cotta and J. I. van Hemert, editors, *EvoCOP*, volume 4446 of *Lecture Notes in Computer Science*, pages 166–175. Springer, 2007.

[14] I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 5(071), 1998.

[15] D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In *COCOON '97: Proceedings of the Third Annual International Conference on Computing and Combinatorics*, pages 251–264, London, UK, 1997. Springer-Verlag.

[16] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, San Jose, CA, USA, 1992.

[17] A. C. Siepel and B. M. E. Moret. Finding an optimal inversion median: Experimental results. In *WABI '01: Proceedings of the First International Workshop on Algorithms in Bioinformatics*, pages 189–203, London, UK, 2001. Springer-Verlag.