Enhancing Parallel Cooperative Trajectory Based Metaheuristics with Path Relinking

Gabriel Luque
Universidad de Málaga, Andalucía Tech
E.T.S.I. Informática, Campus Teatinos,
29071 Málaga (España)
gabriel@lcc.uma.es

Enrique Alba
Universidad de Málaga, Andalucía Tech
E.T.S.I. Informática, Campus Teatinos,
29071 Málaga (España)
eat@lcc.uma.es

ABSTRACT

This paper proposes a novel algorithm combining path relinking with a set of cooperating trajectory based parallel algorithms to yield a new metaheuristic of enhanced search features. Algorithms based on the exploration of the neighborhood of a single solution, like simulated annealing (SA), have offered accurate results for a large number of real-world problems in the past. Because of their trajectory based nature, some advanced models such as the cooperative one are competitive in academic problems, but still show many limitations in addressing large scale instances. In addition, the field of parallel models for trajectory methods has not deeply been studied yet (at least in comparison with parallel population based models). In this work, we propose a new hybrid algorithm which improves cooperative single solution techniques by using path relinking, allowing both to reduce the global execution time and to improve the efficacy of the method. We test here this new model using a large benchmark of instances of two well-known NP-hard problems: MAXSAT and QAP, with competitive results.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; G.1.0 [**Mathematics of Computing**]: General—*Parallel algorithms*

General Terms

Algorithms

Keywords

parallelism, trajectory based metaheuristics, path relinking

1. INTRODUCTION

Metaheuristics are general heuristics that provide suboptimal solutions in a reasonable time for various optimization problems [10]. According to the number of solutions they manage during optimization process, they fall into two categories: trajectory based methods and population based techniques. On the one hand, a population based metaheuristic makes use of a randomly generated population of solutions. The initial population is enhanced through a natural evolution process. At each generation of the process, the whole population or a part of the population is replaced by newly generated individuals (often the best ones). On the other hand, a trajectory based algorithm starts with a single initial solution which, at each step of the search, is replaced by another (often better) solution found in its neighborhood.

Although the use of metaheuristics allows to significantly reduce the temporal complexity of the search process, the exploration remains time-consuming for many industrial and engineering problems. In this context, parallelism emerges as a useful strategy to reduce the computational times down to affordable values. The point is that the parallel versions of metaheuristics allows not only to speed up the computations, but also to improve the quality of the provided solutions [1, 16]. For both trajectory-based and population-based metaheuristics, different parallel models have been proposed in the literature. In general, these parallel models are mostly oriented to study parallel population-based algorithms, but it actually exists a gap in the studies about parallel models for single solution methods from which something could be gained for other researchers.

The focus of this paper is on parallel trajectory-based metaheuristics. Usually, three major parallel models for this kind of algorithms exist: the parallel exploration of the neighborhood, the parallel evaluation of each solution, and the multi-start model. The two first models speed up the execution of the method without changing the semantics of the algorithm in comparison with a sequential exploration. The last one is maybe more interesting from the algorithmic point of view since it can change the behavior of the method with respect to its serial counterpart. The multi-start model lies in launching in parallel several independent or cooperative homo/heterogeneous algorithms. Usually, in its cooperative mode, subalgorithms of the parallel multi-start model exchange information (solutions) during execution and when the target subalgorithm receives a solution, it continues the search using the previous one or the newly received one according to a selection scheme. The problem of this cooperative model is that some interesting information is lost since either the new solution is discarded (it is not chosen by the selection scheme) and no new information is incorporated, or it is accepted and the previous historical information of the subalgorithm is lost.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

ACM 978-1-4503-2662-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2576768.2598337.

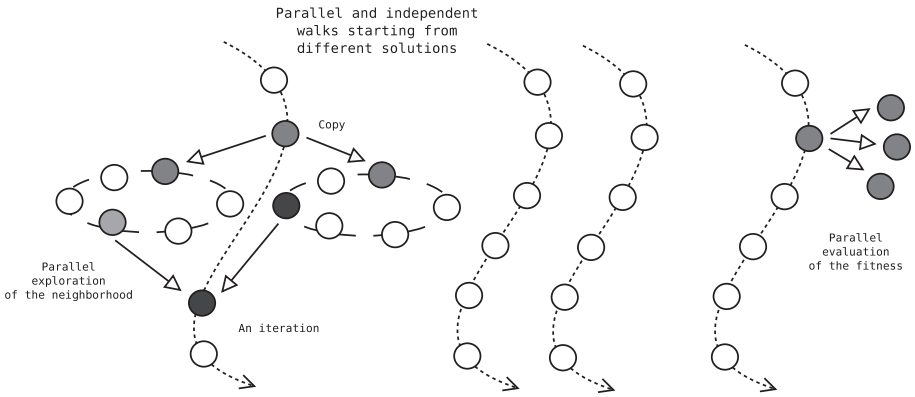

Figure 1: Parallel models for trajectory based methods: parallel exploration of the neighborhood (left), the multi-start model (center), and the parallel evaluation of each solution (right).

This work proposes a new parallel yet simple model that extends the cooperative multi-start model to avoid the aforementioned flaw. In this case, when a solution is received, we use the path relinking technique to generate a set of new candidate solutions which combine the information of the new solution and the previous tentative solution of the target subalgorithm. Therefore, the subalgorithm incorporates new information but, at the same time, it keeps information of its own search. The utilization of path relinking opens some design alternatives: which subalgorithms cooperate, which solution of the set generated by path relinking will be selected to continue the search, ... Then, the main goal of this paper is to propose a new cooperative scheme and study the different design alternatives. This parallel model has been evaluated with two well-known NP-hard problems: MAXSAT and QAP problems [15].

This paper is organized as follows. The next section (Section 2) introduces some background information about parallel trajectory based methods and path relinking. Section 3 presents our proposed parallel model. Then we describe the experimental design used in this paper. Later, we discuss experimental results in Section 5 and finally we summarize the conclusions and give some hints on the future work.

2. BACKGROUND

In this section, we present some background information about the basic techniques that we use to design our new proposal described in this paper.

2.1 Trajectory based techniques

Trajectory based methods, illustrated in Algorithm 1, are single solution-based metaheuristics dedicated to the improvement only one solution in its neighborhood. They start their exploration process from an initial solution randomly generated or provided by another metaheuristic. This solution is then updated during a certain number of steps. At each step, the current solution is replaced by another (often the better) one found in its neighborhood. These methods are mainly characterized by: an internal memory storing the state of the search, a strategy for the selection of the initial solution, a generator of candidate solutions i.e. the neighborhood, and a selection policy of the candidate moves. Three major trajectory based algorithms are largely used: Hill Climbing (HC) [15], Simulated Annealing (SA) [13], and Tabu Search (TS) [11].

Generate($s(0)$);
$t := 0$;
while not Termination_Criterion($s(t)$ **do**
  $m(t) :=$ SelectAcceptableMove($s(t)$);
  $s(t+1) :=$ ApplyMove($m(t)$,$s(t)$);
  $t := t+1$;
end while

Algorithm 1: Trajectory based technique skeleton pseudo-code

Although these technique allows to obtain very accurate results for a large number of problems, some advanced mechanisms have to be used to tackle with the high requirements of the industrial problems. One of these mechanisms is the utilization of parallel models. In the literature are usually identified three major parallel distributed models of this kind of method [2]: the parallel exploration of the neighborhood, the multi-start model, and the parallel evaluation of each solution (see Figure 1).

- **Parallel multi-start model**: The model consists in launching in parallel several independent or cooperative homo/heterogeneous single solution method. Each subalgorithm is often initialized with a different solution. The independent approach is widely exploited because it is natural and easy for the user. In this case, the semantics of the model is the same as the serial execution. That is to say the results obtained with N parallel independent methods is the same as that provided by N algorithms performed in a serial way on a single machine. The parallelism allows to efficiently enhance the robustness of the execution.

  In its cooperative mode, subalgorithms of the parallel multi-start model exchange information during execution. Usually that information is a solution.

- **Parallel exploration of the neighborhood model**: This parallel model is a kind of farmer/worker model allowing to speed up the exploration of the possible

moves without changing the semantics of the algorithm in comparison with a sequential exploration. At the beginning of each iteration of the algorithm, the farmer sends the current solution to a pool of workers. Each worker explores some neighboring candidates, and returns back the results to the farmer.

- **Parallel evaluation of solution model**: The fitness of each solution is evaluated in a parallel centralized way. This kind of parallelism could be efficient if the evaluation function is CPU time-consuming and/or IO intensive.

In the literature, we can found parallel versions of the most popular trajectory based metaheuristics such as parallel SA [8, 4], parallel VNS [17, 7], parallel TS [3, 6], ... But most of them are focused on the application to be solved and they use classical parallel models. The aim of this paper is the parallel model itself, our goal is to provide a new mechanism to build more efficient and accurate parallel solution-based techniques.

The two last models uses the parallel platform to speedup the search procedure but they don't change the behavior of the method. On the contrary, the first parallel model using cooperation modify the dynamics of the technique. In this work, we focus in this kind of methods.

2.2 Path Relinking

Path relinking (PR) [12] was originally proposed into the context of scatter search by extension of its basic philosophy. PR is based on the generation of paths between high quality solutions. This leads to a broader conception of the meaning of creating combinations of solutions. Such combinations may be conceived to arise by generating paths between and beyond selected solutions in neighborhood space, rather than in Euclidean space. This conception is reinforced by the fact that a path between solutions in a neighborhood space will generally yield new solutions that share a significant subset of attributes contained in the parent solutions, in varying "mixes" according to the path selected and the location on the path that determines the solution currently considered. The character of such paths is easily specified by reference to solution attributes that are added, dropped or otherwise modified by the moves executed in neighborhood space. To generate the desired paths, it is only necessary to select moves that perform the following role: upon starting from an initiating solution, the moves must progressively introduce attributes contributed by a guiding solution (or reduce the distance between attributes of the initiating and guiding solutions). In Figure 2, we can observe the scheme followed for this technique to obtain new solutions.

3. OUR PROPOSED MODEL

Our goal is to design a new parallel model for trajectory based metaheuristics which allows to reduce the global execution time but, at the same time, it also improves the efficacy of the exploration of the search space. A number of papers has been devoted to this topic for parallel approaches involving population based methods (some of them also involving trajectory-based ones) but it is not a very studied field for pure parallel trajectory based metaheuristics.

Since we want to improve the efficacy of the resulting parallel algorithm, we focus on the multi-start cooperative paradigm (the other two models do not change the dynamics of the method with respect to the serial version). As discussed in the introduction, a problem in classical approaches of multi-start models for trajectory-based metaheuristics is the lost of information. Indeed, when a subalgorithm receives a solution from other subalgorithm, it has to choose whether it continues the search either with the current one or the newly received one, loosing the stored information in the discarded solution.

We propose a new model in which we do not have to choose between the two solutions, but generate a new solution with the main features of both solutions. With this aim, we can use some mechanisms, similar to the recombination operator of population based method, which combine both solutions [14]. But, in this work, we propose the utilization of a more advanced technique such as path relinking. We run this technique to generate some paths using the current solution and the incoming solution as initial points. The generated path provides the parallel technique of a set of candidate solutions to continue the search, and therefore, a selection scheme is needed to chose one.

Several important design issues arise from the general model proposed in this work:

- **Cooperation scheme:** it indicates what and how subalgorithms cooperate each other.
- **Selection scheme:** it refers what solution is selected from the set of candidate ones generated by PR.

For each design feature we have proposed some alternatives. In the previous existing multi-start models, the features of the incoming solution were not very important rather than it fitness value, but now, this issue can provoke an important impact in the search behaviour. Different possible cooperation schema are analyzed here:

- **Predefined**: in this case, each subalgorithm receives a single solution (the sending island is defined by the topology). Therefore, any subalgorithm only receives a single solution which is combined with the local one.
- **Depending of the fitness value (best)**: in this case, each subalgorithm receives a solution from each subalgorithm which composes the global method. Now, the subalgorithm has to select one solution from this set of candidate solutions, that will be combined with the current one. In this strategy, the selection mechanism is based on the fitness value of the incoming solutions. In this study, we select the solution with the best fitness.
- **Depending of the features of the solution (distance)**: as in the previous one, each subalgorithm receives several solutions (one per subalgorithm) and it has to select one. In this case, the selection will be performed by using a genotypic distance (a diversity measure) among the solutions and we select the farthest one. This distance depends on how the solution are represented in the algorithm.
- **Random**: as in the previous scenarios, each subalgorithm receives several solutions (one per subalgorithm) but in this strategy, random one is selected from all the incoming solutions.

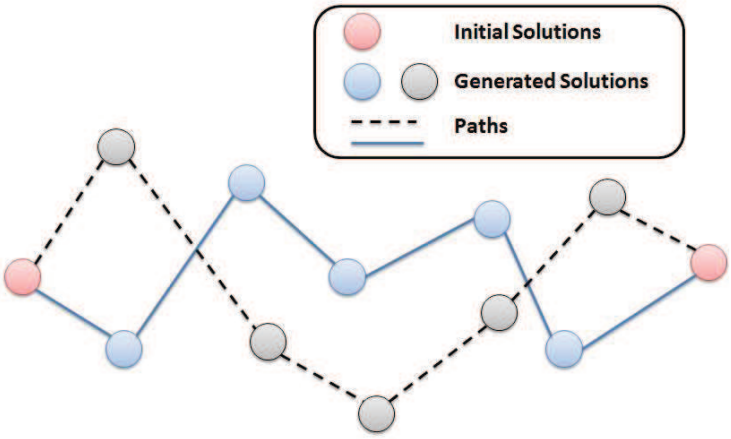

Figure 2: Path relinking scheme.

Using one of the these strategies, our proposed technique obtains the second initial solution (the first initial one is the current point of the subalgorithm) and then, we can apply the path relinking to generate a path. That path provides some new candidate solutions and the method has to select one of them to replace the current one and continue the search process. To choose the new solution we also take into account some alternatives:

- **Best**: the algorithm selects the best solution in the generated path:

$$max_{s \in S}(f(s)), \tag{1}$$

  where S is a set of solution composed by the points visited during the path generated by PR and f is the fitness function (assuming the maximization case).

- **Most shared information**: in this case, the subalgorithm selects the solution sharing more information from initial solutions. To do this, we calculate the genotypic distance among the solutions and then we apply the next equation:

$$min_{s \in S}(max(dist(s, s'), dist(s, s''))), \tag{2}$$

  where S is a set of solution composed by the points visited during the path generated by PR, s' and s'' are the initial solutions and *dist* is the genotypic distance between two solutions. With this process we simultaneously minimize the distance of the new solution with respect to the two initial ones.

- **Random**: in this case, the subalgorithm select a random solution from the path without taking into account its quality or any other feature.

In the experimental section we study the behaviour and the performance of each strategy.

4. EXPERIMENTAL DESIGN

In this section, we describe the experimental design followed in this work. First, we present the problems and the instances used in the experiments. Later, we explain the algorithm used to test our parallel model. Finally, we present the methodology and parameters used in the experiments.

4.1 Benchmark

In order to make more relevant contribution, we have selected a wide set of instances from two very different problems: MAXSAT and QAP.

MAXSAT: The satisfiability (SAT) problem is commonly recognized as a fundamental problem in computer science, automated reasoning, mathematical logic, and related fields. The MAXSAT is a variant of this general problem.

Formally, the SAT problem can be formulated as follows. Let $U = \{u_1, \ldots, u_n\}$ be a set of n Boolean variables. A truth assignment for U is a function $t : U \rightarrow \{true, false\}$. Two literals, u and $\neg u$, can match with each variable. A literal u (resp. $\neg u$) is true under t if and only if $t(u) = true$ (resp. $t(\neg u) = false$). A set C of literals is called a clause and it represents the disjunction (*or* logical connective). A set of clauses is called a formula. A formula f is interpreted as a formula of the propositional calculus in conjunctive normal form (CNF) so that a truth assignment t satisfies a clause C iff at least one literal $u \in C$ is true under t. Finally, t satisfies f iff it satisfies every clause in f. The SAT problem consists of a set of n variables $\{u_1, \ldots, u_n\}$ and a set of m clauses $C_1, \ldots, C_m$. The goal is to determine whether there exists or not an assignment of truth values to variables that makes the formula $f = C_1 \wedge \cdots \wedge C_m$ in CNF satisfiable. Among the extensions to SAT, MAXSAT [9] is probably the most known one. In this case, a parameter K is given and the problem is to determine whether there exists an assignment t of truth values to variables such that at least K clauses are satisfied. SAT can be considered as a special case of MAXSAT when K equals the number m of clauses.

In the experiments we use five instances from standard benchmark SATLIB[1]. We use the instances: *uf250-01.cnf* - *uf250-05.cnf*. Each instance has 250 variables and 1065 clauses, and all them are in the transition phase.

QAP: The Quadratic Assignment Problem (QAP) is a well-known NP-hard combinatorial optimization problem, which is the core of many real-world optimization problems [9]. QAP models many applications in diverse areas. In par-

[1] `http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`

ticular the following problems can be formulated as special cases of the QAP: the Traveling Salesman Problem (TSP), the Fragment Assembly Problem (FAP), Scheduling problems, the Facility Layout Problem, among others.

Let P be a set of n facilities and L a set of n locations. For each pair of locations i and j, an arbitrary distance is specified r_{ij} and for each pair of facilities p and q, a flow is specified w_{pq}. The QAP consists of assigning to each location in L one facility in P in such manner that the total cost of the assignment is minimized. Each location can only contain one facility and all facilities must be assigned to one location. For each pair of locations, the cost is calculated as the product of the distance between the locations and the flow associated with the facilities in the locations. The total cost is the sum of all the costs associated with each pair of locations. One solution to this problem is a bijection between L and P, that is, $x : L \rightarrow P$ such that x is bijective. Without loss of generality, we can just assume that $L = P = \{1, 2, ..., n\}$ and each solution x is a permutation in S_n, the set permutations of $\{1, 2, ..., n\}$.

The cost function to be minimized can be formally defined as:

$$f(x) = \sum_{i,j=1}^{n} r_{i,j} \cdot w_{x(i),x(j)} \quad (3)$$

We chose the five most complex QAP instances of QAPLIB Library[2]. The complexity of the QAP instances is given by their size and autocorrelation length ℓ [5]. In consequence, we selected esc128, tho150, tai100b, tai150b, and tai256c instances that are described in Table 1.

Table 1: The five most complex instances of QAPLIB.

Instance	*Size*	ξ	ℓ	*Lowest Known Cost*
esc128	128	32	32	64
tho150	150	41.19	44.174	8.13E+06
tai100b	100	35.472	39.613	1.19E+09
tai150b	150	40.458	42.947	4.99E+08
tai256c	256	64	64	4.48E+07

4.2 Algorithm

We have used the well-known simulated annealing algorithm to test our parallel model. Simulated annealing (SA) [13] is a stochastic optimization technique, which has its origin in statistical mechanics. It is based upon a cooling procedure used in industry. This procedure heats the material to a high temperature so that it becomes a liquid and the atoms can move relatively freely. The temperature is then slowly lowered so that at each temperature the atoms can move enough to begin adopting the most stable configuration. In principle, if the material is cooled slowly enough, the atoms are able to reach the most stable (optimum) configuration. This smooth cooling process is known as *annealing*. Algorithm 2 shows a scheme of SA. First at all, the parameter T, called the temperature, and the solution, are initialized. The solution $s1$ is accepted as the new current solution if $\delta = f(s1) - f(s0) < 0$. Stagnations in local optimum are prevented by accepting also solutions which increase the objective function value with a probability $exp(-\delta/T)$ if $\delta > 0$. This process is repeated several times to obtain good sampling statistics for the current temperature. The number of such iterations is given by the parameter $Markov\_Chain\_length$, whose name alludes the fact that the sequence of accepted solutions is a Markov chain (a sequence of states in which each state only depends on the previous one). Then the temperature is decremented and the entire process repeated until a frozen state is achieved at T_{min}. The value of T usually varies from a relatively large value to a small value close to zero.

```
t = 0
initialize(T)
s0 = Initial_Solution()
v0 = Evaluate(s0)
repeat
  repeat
    t = t + 1
    s1 = Generate(s0,T)
    v1 = Evaluate(s0,T)
    if Accept(v0,v1,T) then
      s0 = s1
      v0 = v1
    end if
  until t mod Markov_Chain_length == 0
  T = Update(T)
until 'loop stop criterion' satisfied
```

Algorithm 2: Scheme of the Simulated Annealing(SA) Algorithm.

In order to apply SA to solve our problems, we have to define how a solution in the neighborhood is generated (function Generate in Algorithm 2). Since we use a bitstring to represent a solution for MAXSAT, we use the standard bit-flip operator (it change a single bit from 0 to 1 or from 1 to 0) to generate a neighbor from the current solution. In the QAP case, we use an integer permutation in order to represent a solution inside the algorithm and the swap operator (the values of two positions are interchanged) is the selected mechanism to build a new solution from the current one.

In our parallel approach we run eight independent instances (subalgorithms) of SA which asynchronously cooperate every 50 iterations. As cooperation scheme, we use the four alternatives presented in Section 3. When a new solution arrives to the target subalgorithm, the PR method is applied to generate a path. To build this path, first, it analyzes the component which are different between the two initial solutions, and then, each visited point in the path is generated by changing one of these components.

This way of building the path allows us to obtain efficient implementations of the selection scheme (see Section 3). For example, we do not need to build the whole path when we want to select a random solution from the path or when we plan to use a solution which maximizes the shared information, since they can be calculated *a priori* only generating a single solution. However, to obtain the best solution of the path, all the visited points should be generated and evaluated. In order to the reduce the computation cost of this last strategy, we will use partial evaluations instead of a complete evaluation of the solution. This allows to reduce the computational cost of the resulting algorithm. For example,

[2] `http://www.seas.upenn.edu/qaplib/`

in the MAXSAT problem, when the variation operator only modify a single logic variable, we only have to analyze the effect on a low percentage (2-5%) of clauses in the instance. Also, in the QAP problem, we can reduce the computational complexity of the evaluation of a solution from $O(n^2)$ (a complete evaluation) to $O(n)$ (when a partial evaluation is performed).

4.3 Methodology

This subsection provides the reader with the details of the experiments performed to evaluate the new parallel model proposed for trajectory-based metaheuristics. We have analyzed 12 different variants (three selection strategies and four cooperation schema). We use the terminology **SA_X_Y**, where **X** is the selection mechanism and **Y** is the cooperation strategy. The possible values for **X** are: **rnd** for random, **bst** for best, and **inf** for maximizing the shared information scheme. The possible values for **Y** are: **pre** for the predefined topology, **bst** for the best solution, **dst** for the strategy based on the distance, and **rnd** for the random one. We will also compare our proposed model with a parallel version using the multi-start no-cooperative model, also known as independent run model (**iSA**) , and a parallel version using the classical multi-start cooperative model (**cSA**), in which incoming solutions just replace the current one. In order to perform fair comparisons, the stopping condition is to find the optimal solution.

The experiments have been executed on a Intel Pentium IV 2.8GHz with 512MB running SuSE Linux 8.1. Because of the stochastic nature of the algorithms, we perform 30 independent runs of each test to gather meaningful experimental data and apply statistical confidence metrics to validate our results. First, we use the Kolmogorov-Smirnov test to check whether the data follows a normal distribution or not. If so, then we carry out an ANOVA test to compare the means; otherwise, a Kruskal-Wallis test is used to compare the medians. In each case, a confidence level of 99 % is used.

5. ANALYSIS OF THE EXPERIMENTS

In this section we analyze the results of the different variants of our proposed model. First, we study the accuracy of the methods and then, we discuss their computational cost.

5.1 Accuracy

Let's first compare the accuracy of the different algorithms. Since there are many different problem instances and analyzing them thoroughly would hinder us from drawing clear conclusions, we have summarized in Table 2 the results. In this table we only study the accuracy of the techniques. Since the stopping criterion is to find the optimal solution (if it is possible since some variants get stuck in a local one), to measure the accuracy we use the number of instances solved by the method (the algorithm was able to find the optimum). We use two different values: the first one is the number of instances in which the algorithm found the optimal solution in at least one run; and the second one is the number of instances in which the algorithm found the optimal solution in *robust* way (this means the algorithm find the optimum in at least 25 out 30 independent runs). The range of both values is between 0 and 10 (5 instances of QAP plus 5 instances of MAXSAT)

From Table 2, we can obtain several interesting conclusions. First, we can note that cooperative schema significantly outperform to non-cooperative ones. In fact, **iSA** is not able to find the solution to any instance in a robust way. A second important conclusion is that all the variants of our model outperform traditional parallel models for trajectory based methods. This results shows that the exploration scheme induced by our model is more accurate than the other parallel algorithms in the context of this problem.

Cooperation scheme	Selection scheme		
	bst	**rnd**	**inf**
pre	9 - 5	7 - 3	8 - 3
bst	**10 - 10**	**10** - 6	**10** - 8
rnd	**10** - 6	8 - 3	**10** - 4
dst	**10 - 10**	9 - 4	**10** - 8
iSA		5 - 0	
cSA		7 - 2	

Table 2: Accuracy of the algorithms.

Analyzing the different variants of the proposed model, it can be seen that the models that make use of some information from the incoming solutions (fitness, distance or shared information) outperform the variants which are based on other features (random or predefined topologies). This is an expected result since the utilization of additional information during the process allows the method to have more elements to guide its search.

In concrete, we can see that cooperation strategies using the best incoming solution or the farthest solution are the best variants. This is a quite surprising result, since these techniques promote very different search behaviours (the **bst** strategy favors the intensification while the **dst** scheme promotes the diversification) but both methods get very high-quality solutions. A similar behaviour can be observe when we analyze the different selection method: the **bst** (which promotes intensification) and **inf** (which promotes diversification) strategies obtain equivalent results. Although the result are similar, we can notice a clear trend toward the techniques which favor the intensification. In fact, **SA_bst_bst** and **SA_dst_bst** are the only ones which can solve all the instances in all the runs.

5.2 Computational cost

Now, we focus on the computational cost: numerical performance (number of partial evaluations) and wall-clock time (in seconds). In order to perform a fair comparison, we only consider the algorithms which get similar results. In concrete, we compare **SA_bst_bst**, **SA_bst_inf**, **SA_dst_bst** y **SA_dst_inf** using the instances which are robustly solved by these methods. In Figure 3, we show the numerical performance (left figure) and the runtime (right figure). Both values are normalized with respect to the value obtained by **SA_dst_inf**.

From Figure 3, we can distinguish three different behaviours according to the statistical analysis (all the results are statistically different with the exception of **SA_X_inf** models) . The first behaviour is the presented by **SA_dst_bst** which is the variant with the highest computational cost. This is expected since the initial solutions selected by **dst** cooperation strategy are very different and therefore the generated path are longer and it also has to evaluate all the generated solution to find the best solution in the path (**bst** selection scheme). The second behaviour

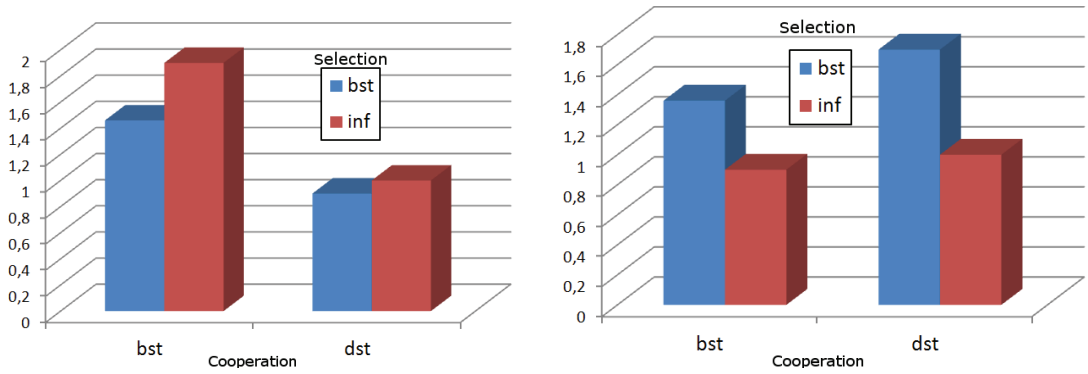

Figure 3: (a) Numerical performance (number of partial evaluations) and (b) execution time (seconds).

is the provoked by **SA_bst_bst** which is the second variant with a higher computational cost. This high cost is due to the utilization of **bst** selection scheme, in which the variant has to evaluate all the solutions in the path, although in this case the path are usually shorter than in the previous variant (**SA_dst_bst**). Finally, the last behaviour is the presented by **SA_X_inf** strategies which need less runtime to find the solution. This is mainly due to the **inf** selection scheme only need to evaluate a single solution in the path.

Summarizing we can conclude that variants using the best solution in the path obtain the best accurate results but they need more execution time to find these high-quality solutions. By contrast, strategies using the shared information between solutions are faster but the solution obtained are slightly worse.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we have developed a new parallel model for trajectory based methods, which improves the cooperation phase by means of adding path relinking technique. The utilization of this last technique allow to generate a wide set of candidate solutions to continue the search. This set is composed by solutions which include information from the current solution of the subalgorithm and also information from the incoming solution.

The results show that our proposed method is more accurate and efficient than the existing one. We have studied different design alternatives such as the several cooperation schema or different mechanism to select the next solution from the set of solutions generated by path relinking. Each variant has its own advantages and drawbacks. For example, we saw that using the best solution of the path, the algorithm obtains very accurate solution but the computation cost is higher.

As future work, we plan to extend this study to other problems or other trajectory based methods for generalizing the conclusion of this paper. In this paper, we have observed that the evaluation of the point visited by the path generated by PR is a quite high-consuming process, then we want to analyze different alternatives to perform that process (maybe using some theoretical results about the search space) and then speed up the search.

Acknowledgments

The authors acknowledge funds from the Spanish Ministry of Sciences and Innovation European FEDER, under contract TIN2011-28194 (roadME http://roadme.lcc.uma.es), Andalucía Tech, and from the project number 8.06/5.47.4142 in collaboration with the VSB-Technical University of Ostrava.

7. REFERENCES

[1] E. Alba, editor. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, 2005.

[2] E. Alba, G. Luque, and S. Nesmachnow. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48, 2013.

[3] W. Bożejko, J. Pempera, and C. Smutnicki. Parallel tabu search algorithm for the hybrid flow shop problem. *Computers & Industrial Engineering*, 65(3):466–474, 2013.

[4] Y.-L. Chang, K.-S. Chen, B. Huang, W.-Y. Chang, J. A. Benediktsson, and L. Chang. A parallel simulated annealing approach to band selection for high-dimensional remote sensing images. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 4(3):579–590, 2011.

[5] F. Chicano, G. Luque, and E. Alba. Autocorrelation measures for the quadratiec assignment problem. *Applied Mathematics Letters*, 25(4):698–705, 2012.

[6] J.-F. Cordeau and M. Maischberger. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, 39(9):2033–2050, 2012.

[7] M. Eskandarpour, S. H. Zegordi, and E. Nikbakhsh. A parallel variable neighborhood search for the multi-objective sustainable post-sales network design problem. *International Journal of Production Economics*, 145(1):117–131, 2013.

[8] A. Ferreiro, J. García, J. López-Salas, and C. Vázquez. An efficient implementation of parallel simulated annealing algorithm in gpus. *Journal of Global Optimization*, 57(3):863–890, 2013.

[9] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[10] M. Gendreau. *Handbook of metaheuristics*, volume 146. Springer, 2010.

[11] F. Glover. Tabu Search, part I. *ORSA, Journal of Computing*, (1):190–206, 1989.

[12] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 39(3):653–684, 2000.

[13] S. Kirkpatrick, C. Gellatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[14] G. Luque, F. Luna, and E. Alba. A new parallel cooperative model for trajectory based metaheuristics. In *Distributed Computing and Artificial Intelligence*, pages 559–567. Springer, 2010.

[15] C. Papadimitriou. *The Complexity of Combinatorial Optimization Problems*. Master's thesis, Princeton University, 1976.

[16] E.-G. Talbi. *Parallel combinatorial optimization*, volume 58. John Wiley & Sons, 2006.

[17] M. Yazdani, M. Amiri, and M. Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37(1):678–687, 2010.