Autonomous Control Approach for Local Search

Julien Robet, Frédéric Lardeux, Frédéric Saubion

An Autonomous Control Approach for Local Search

Julien Robet, Frédéric Lardeux, and Frédéric Saubion

LERIA, University of Angers (France)
name@info.univ-angers.fr

1 Introduction

Local search algorithms are metaheuristics which have been widely used for solving complex combinatorial problems. Their efficiency relies on their ability to suitably explore various areas of the search space but also on its propensity to converge to a local optimum (the locality is defined here with respect to the notion of neighborhood). The concept of balance between intensification and diversification, especially well-known in evolutionary computation, is a crucial point when designing and using a local search algorithm. Indeed, one of the classic pitfalls encountered by these algorithms is the excessive attraction of local optima, which may trap the search process when all the potential neighbors are not as good as the current configuration and when the move strategy is mainly based on improvement. To cope with this excessive exploitation of the search space (i.e., intensification), alternative mechanisms must be used to insure enough diversification.

Inspired by the recent book of R. Battiti et al.[1] and our previous work on the autonomous management of multiple operators in genetic algorithms [2], we propose an original approach in order to design a local search algorithm that will include several move operators, corresponding to different neighborhoods and different strategies for choosing the neighbors. The control of these operators will then be achieved automatically. We have tested our algorithm on the famous quadratic assignment problem (QAP), which has been widely studied and for which an extensive library of instances and results is available [3].

2 Toward a more Integrated View of Neighborhood

In this paper, our idea is to combine parameters and components in the notion of move operators and to automatically control their application along the search process. We therefore introduce, within the local search algorithm, an adaptive operator selection method, as we have already proposed for genetic algorithms [2]. This selection mainly consists in evaluating the effect of the operator on the current state of the search in order to reward them and to be able to choose the most suitable one for the next computation step. Therefore, our objective is to evaluate the impact of the operators and to adjust their use according to the current the search. This approach is summarized in Figure 1.

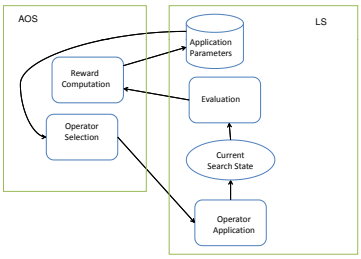

Fig. 1. Autonomous control in local search

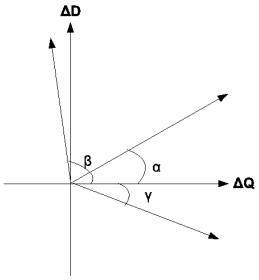

Fig. 2. Balance between intensification and diversification

This figure allows us to highlight the main issues that we have to address in an autonomous local search algorithm:

- How to evaluate the current search state ?
- How to reward operators with regards to this evaluation ?
- How to use these rewards to select the operator for the next move ?

If the notion of quality helps to guide and evaluate the ongoing search process, the concept of diversity should also deserves more attention. It has appeared indeed in previous works [2] that this notion can be used jointly with the quality in order to efficiently manage the balance between diversification and intensification. In the next section, we will therefore propose a definition of a diversity measure according to local search specificities.

3 A Diversity Measure for Local Search

To quantify the diversity of a local search path, our measure consists in analyzing, for each variable of the problem , the distribution of its successive assigned values with regards to its domain. Our approach relies indeed on a simple observation: if a search path is very diversified, successive assigned values to variables will tend to "cover" their domains. Otherwise, a non diversified path will be characterized by assigning a small number of different values to each variable (with regards to the cardinal of its domain), especially for the variable whose value remains unchanged along the considered path. The evaluation consists in observing, for each decision variable, the standard deviation of the number of occurrences of each possible value for this variable (i.e., for each element of its domain). The average of these deviations assesses the intrinsic similarity of the path. Indeed, the calculated standard deviations will be even lower as the path will be diverse. In order to normalize the measure, this average value is then divided by the theoretically maximum possible similarity. We obtain then a value between 0 and 1 that we subtract from 1 to reflect the diversity of the path.

4 The ALS Algorithm

The aim of our algorithm, called ALS (Autonomous Local Search), is to manage a set of local search operators in order to apply them at appropriate moment during the solving process. The challenge is thus to make three main modules work together: current solving state evaluation, internal components rewarding, and selection of the next operator using these rewards.

4.1 Operator Evaluation

The process used to analyze internal components is inspired by our previous works on evolutionary algorithms [2]. Its principle is to maintain along the search a history of recent performances for each operator. The originality of the method relies in the fact that the evaluation is not limited to one criterion (quality variations), but also takes into account diversity gaps. Quality and diversity variations between two iterations are thus computed as:

$$\Delta Q = \frac{eval(op(c)) - eval(c)}{eval(c) + 1} \text{ and } \Delta D = div(P_{i,j}) - div(P_{i-1,j-1})$$

where $op(c)$ is the configuration produced by the application of op on the current configuration c and $P_{i,j}$ the path from iteration i to iteration j. Given an operator op we then define $\Delta Q_{op,t}$ (resp. $\Delta D_{op,t}$) the mean quality (resp. diversity) variation over the t last applications of op where t corresponds to the size of the sliding window that stores information about each operator.

4.2 Control Issues

Applying the same memorization principles to the variations of the search state, we may collect important information about how the search process evolves among the search space. Indeed, a diversity loss reflects a focus on a particular search space areas, whereas a diversity gain appears when moving away from the current area. During the solving process, the choice of next operator to apply is achieved according to probabilities defined in the following section, 4.3. These probabilities are widely influenced by a parameter α, which models the desired balance between intensification and diversification. We thus introduce three values (see figure 2):

- α corresponds to the desired balance between intensification and diversification. ($0 \leq \alpha \leq \pi/2$)
- β is the angle formed by the current search trajectory (actual current angle)
- γ is the resulting commanded angle for next search step, in order to counteract the search in the right direction

The application strategy can thus been seen as the way to compute γ's value according to collected information. For example, if one gets trapped into a local

optima, it will be beneficial to increase γ in order to promote search diversification. Furthermore, the closest to α the search trajectory is, the more efficient the solving process is. We designed then a formula to compute γ according to α and β, reducing the gaps between them as much as possible:

$$\gamma = \begin{cases} \alpha - gap(\alpha, \beta)/2 & \text{if } gap(\alpha, \beta) \leq \pi/2 \\ \alpha - gap(\alpha, \beta + \pi)/2 & \text{otherwise} \end{cases}$$

where *gap* corresponds to the difference between two angles values.

4.3 Operator Rewarding

Relying on operator evaluation (cf. section 4.1) and current execution state (cf. section 4.2), we defined the following rewarding system: first of all, measures introduced in section 4.1 have to be normalized. We thus divide them by the highest absolute values found among all operators.

Then, for each operator *op*, we have to compute its corresponding $angle(op)$ (between 0 and 2π) and its norm $||op||$ (between 0 and $\sqrt{2}$, as measures are normalized) in (Δ_Q, Δ_D). Operator rewards are then defined as follows:

$$score(op) = \begin{cases} (||op||.(1 - \frac{4.gap(\gamma, angle(op))}{\pi}))^2 & \text{if } gap(\gamma, angle(op)) \leq \pi/4 \\ (||op||.\frac{\pi - 4.gap(\gamma, angle(op))}{3\pi})^2 & \text{otherwise} \end{cases}$$

Those rewards are finally used to define operators' application probabilities:

$$p(op_k) = \begin{cases} max(0, \frac{score(op_k)}{\Sigma_{o \in Op} score(o)}) & \text{if } \exists o \in Op, score(o) > 0 \\ \\ \frac{[\Sigma_{o \in Op} score(o)] - score(op_k)}{\Sigma_{o \in Op} score(o)} & \text{otherwise} \end{cases}$$

where Op is the set of all possible operators.

In order to insure fairness for less used operators, a simulation step is achieved every 20 iterations. It consists in applying every operator on the current configuration, only keeping the best resulting combination. This method, although computationally expensive, allows relatively up-to-date evaluations.

5 Application to the Quadratic Assignment Problem

We have experimented ALS on instances from QAPLIB [3]. We consider ALS with 10 operators , and tried several values for α. As a baseline, we have implemented an uniform choice version (at each step, an operator is chosen randomly). ALS is also compared against the optimized robust taboo search [4] with the same experimental conditions. Table 5 summarizes the results: for each instance, the best known value (March 2009) is indicated in the BKV column, θ_{avg} is then the mean deviation from BKV, computed as $\theta_{avg} = \frac{100(f_{avg} - BKV)}{BKV}$, where f_{avg} stands for objective function's mean value over 20 runs, each of them

Instance	BKV	UC	ALS				RTS
			$\alpha = 0.25\pi$	$\alpha = 0.15\pi$	$\alpha = 0.1\pi$	$\alpha = 0$	
bur26a	5426670	0,1177	0,0196	0,0020	**0,0000**	0,0020	**0,0000**
bur26c	5426795	0,0359	0,0029	**0,0000**	**0,0000**	0,0217	**0,0000**
bur26f	3782044	0,0153	0,0019	**0,0000**	**0,0000**	0,0679	**0,0000**
chr25a	3796	42,4341	10,2160	**6,6228**	8,3298	8,6828	7,6765
els19	1,7E+07	4,8532	0,0003	**0,0000**	**0,0000**	**0,0000**	**0,0000**
kra30a	88900	3,8774	0,8931	0,7627	0,4027	1,1755	**0,0000**
kra30b	91420	2,4251	0,3227	**0,0131**	0,0459	0,1181	0,0230
tai30b	6,4E+08	1,5794	0,1882	0,2319	0,0372	0,8525	**0,0326**
tai50b	4,6E+08	1,4307	0,2330	0,3693	0,2566	0,5376	**0,1078**
nug20	2570	1,9767	0,0156	**0,0000**	**0,0000**	**0,0000**	**0,0000**
nug30	6124	2,0901	0,2449	**0,0000**	**0,0000**	0,0131	0,0065
sko42	15812	2,0529	0,5237	0,0443	**0,0202**	0,0620	0,0342
sko49	23386	2,0174	0,6431	0,2279	0,2407	0,2382	**0,1403**
sko56	34458	2,1139	0,5305	0,1843	0,1660	0,2783	**0,1051**
tai30a	1818146	3,6971	1,7781	0,4163	0,6008	0,3973	**0,3933**
tai35a	2422002	3,9348	2,1099	0,8157	**0,6868**	0,9082	0,7705
tai50a	4941410	4,4437	2,4926	1,1522	**1,1269**	1,3648	1,3733
wil50	48816	1,0005	0,2176	0,0520	0,0385	0,0713	**0,0361**
Average		4,4498	1,1352	0,6053	0,6640	0,8217	**0,5944**

Table 1. Mean deviation of ALS, uniform choice, and robust taboo search from best known values (BKV)

being executed over 40000 iterations. The best value between the five algorithms is boldfaced. Basically, results from table 5 clearly highlights the controller's advantages, since among 18 tested instances, ALS is systematically better than UC. When compared to RTS, we may remark that with α set to 0.15π or 0.1π, we obtain very similar results since about half of the instances are better solved with ALS. Nevertheless, we should insist here on the fact that the 10 operators are not really optimized and could be seen as general purpose operator for permutation based coded problems. The average values, mentioned at the bottom of the table and which are not so different for α between 0 and 0.15π, highlights that the tuning of α, although having a noticeable impact, seems to bear a greater tolerance than the tuning of all the operators' parameters. Let us also mention that according to other tests, not reported here, enlarging the set of handled operators improves solving efficiency. Indeed, ALS-10 (10 operators set) was better than ALS-2 (2 operators set) for all the instances. Therefore, using our controller, the user may expect benefits from a multi-operators algorithm whose parameters would be actually difficult to tune with regards to their combinatorial interactions. The controller aims at applying the right operator at the right time, even if some of them could be judged a priori less interesting

References

1. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Vol 45 of Operations Research/Computer Science Interfaces. Springer Verlag (2008)
2. Maturana, J., Fialho, A., Saubion, F., Schoenauer, M., Sebag, M.: Compass and dynamic multi-armed bandits for adaptive operator selection. In: Proceedings of IEEE Congress on Evolutionary Computation CEC. (2009) to appear.
3. Burkard, R.E., Karisch, S., Rendl, F.: Qaplib-a quadratic assignment problem library. European Journal of Operational Research **55**(1) (November 1991) 115–119
4. Taillard, É.: Robust taboo search for the quadratic assignment problem. Parallel Computing **17**(4-5) (1991) 443–455