A Tabu Search Method for Interval Constraints

Charlotte Truchet[1], Marc Christie[2], and Jean-Marie Normand[1]

[1] LINA, UMR 6241,
Université de Nantes, 2 rue de la Houssinière, 44322 Nantes, France
[2] IRISA/INRIA Rennes - Bretagne Atlantique, Campus de Beaulieu, 35042, Rennes, France
{charlotte.truchet,jean-marie.normand}@univ-nantes.fr,
marc.christie@irisa.fr

Abstract. This article presents an extension of the Tabu Search (TS) metaheuristic to continuous CSPs, where the domains are represented by floating point-bounded intervals. This leads to redefine the usual TS operators to take into account the special features of interval constraints: real variables encoded in floating points domains, high cardinality of the domains, nature of the CSP where constraints may be partially satisfied. To illustrate the expressiveness of the framework, we instantiate this method to compute an inner-approximation of a set of inequalities.

Metaheuristics, in particular Tabu Search (TS, [6,8]), have largely proven their efficiency on discrete optimization or constraint problems. In this article, we propose a framework to extend TS to continous problems on real variables. There exist a variety of optimization techniques to solve such problems, but they are usually dedicated to particular types of constraints (linear, polynomial, differentiable) and do not offer the guarantees of interval approaches [4,5]. Interval-based solvers like Realpaver [7] are dedicated to compute rigourous inner and outer-approximations of continuous problems, but often fail in efficiently computing a first solution due to the complete nature of the search process. We provide a unified way to express continuous CSPs in a TS framework that shares the reliability of interval techniques, and tackle a problem not well adressed by classical tools: the computation of a single solution for interval CSPs or optimisation problems, whatever the type of the constraints.

1 Interval Constraints

Interval arithmetics [9] offers a reliable solution to avoid rounding errors due to finite representation of real values (see IEEE754 norm). Real values are encompassed within *floating-point intervals*: a real value r may be represented by any interval I, with floating-point bounds, containing r. The set of all intervals is $\mathbb{I}$. The Cartesian product of intervals is called a *box*. The classical operators over $\mathbb{R}$ can be redefined over $\mathbb{I}$ by enforcing the fundamental property of containment[9]: the interval extension of a binary operator $\square$ is given by the smallest interval containing the results of the application of $\square$. This construct

L. Perron and M. Trick (Eds.): CPAIOR 2008, LNCS 5015, pp. 372–376, 2008.

guarantees that no values are lost, but can lead to an over-estimation of the result. Real functions are extended on $\mathbb{I}$ in the same way.

A CSP defined over continuous domains is translated into intervals by taking interval variables over interval domains. The constraints are extended over $\mathbb{I}$ in the same way as functions or operators. The goal is to find a box $\boldsymbol{B} \subseteq D_1 \times \ldots \times D_n$ such that the constraints are satisfied. An interval constraint C on a box $\boldsymbol{B}$ can be *certainly satisfied* (every real vector in $\boldsymbol{B}$ satisfies C), *certainly not satisfied* (no real vector in $\boldsymbol{B}$ satisfies C). If neither of those two properties can be computed, C is said *partially satisfied*.

Several metaheuristics have been transposed to CSP on real variables. Some of them are adapted from TS [4,5,2]. However, these algorithms are based on real configurations, and employed to tackle optimization problems. Our method is closer to that of [3] for interval CSPs with non-linear differentiable constraints, or to [1] on multi-objective problems.

2 Tabu Search on Intervals

The basic components of a TS algorithm on discrete domains are well-known: a penalty function $f : D_1 \times \ldots \times D_n \rightarrow \mathbb{N}$, counting the number of violated constraints most of the times, a neighborhood function, and the exact definition of a tabu mechanism (basic tabu behaviour with size `t`, dynamic tabu, aspiration, ...). However, adapting the algorithm to new types of domains leads to a number of obstacles. Firstly, TS algorithms use many operators which are trivial over discrete configurations (equality, membership, random choice) but require specific attention over interval domains. Secondly, an interval CSP has a search space with specific properties: the domains are huge, and they are defined by two floating point numbers but include an infinite set of reals. Hence we dissect the algorithm into more precise atomic components.

2.1 Framework

Random on $\mathcal{S}$. A random function `random` : $\mathcal{S} \rightarrow \mathcal{S}$ is needed both for the random restart, and to choose a neighbor of a current configuration. An interval is characterized by its center, length and bounds. A uniform random law cannot be easily defined over these parameters; choosing the center (or one bound) first leads to a bias toward small intervals, and conversely, choosing the length first leads to bias toward centered intervals. To both ensure diversification and limit the number of possibly satisfied intervals, we choose the center first, and then the length L. L follows a repartition law $P(L \leq z) = 2b*(1-ln(2z/b-a))/(b-a)$ with an average of $(b-a)/8$.

Neighborhood exploration. A neighborhood is a subset of $\mathcal{S}$, supposedly close to the current configuration either in terms of Euclidean distance in the domain space of in terms of solution similarity. We divide this into two steps. A first operator `neighborhood` : $\mathcal{S} \rightarrow 2^{\mathcal{S}}$ computes a neighbor area, based on the same

principle than on discrete problems. Choosing configurations at a Hamming distance of 1, we can define for instance $\texttt{neighborhood}_1(< v_1 \dots v_j \dots v_n >) = \{\forall v'_j \subset D_j, < v_1 \dots v'_j \dots v_n >\}$, which tries to move a randomly chosen variable, or $\texttt{neighborhood}_2 = \{\forall 1 \leq j \leq n, \forall v'_j \subset D_j, < v_1 \dots v'_j \dots v_n >\}$ which moves every variable once.

Sampling the neighborhood. The second neighborhood operator is in charge of the neighbourhood sampling, which has to be done partially because of the cardinality of the results of `neighborhood`. This requests that $\mathcal{S}$ comes with an operator $\texttt{sample} : \mathcal{S} \to 2^{\mathcal{S}}$, to choose a good sample of boxes within a given box. This function is parameterized by the size of the sample, nb_{neigh}. We propose two possibilities for `sample`. The first one is merely random: let $\texttt{sample}_1(B)$ be a set of nb_{neigh} random configurations, with calls to `random`. The second one ensures that the box B is widely explored: $\texttt{sample}_2$ consists in cutting the box B into nb_{neigh} equal parts and choosing randomly a configuration in each part.

Penalty function. Technically, a penalty function needs to be computable in a reasonable time (if possible, incrementally), to have comparable values, and express the satisfaction of the problem. One can measure the satisfaction of a problem by counting the number of satisfied constraints. However this raises a problem on interval CSPs due to interval approximation: each configuration is either satisfied, non satisfied or partially satisfied, which questions the way to encompass the partially satisfied constaints in the count. The definition of f influences the semantic of the problem. In the problem we consider, we have focused on inner approximations of the solution set. The penalty function is thus defined as $f(s) = ps(s) + cns(s)$, where $ps(s)$ (resp. $cns(s)$) represents the cardinality of the partially satisfied constraints (resp. non-satisfied), for a configuration s. In such a case, $f(s) = 0$ iff $ps(s) + cns(s) = 0$, iff $cs(s) = p$, that is, all the constraints are satisfied by s.

Tabu mechanism. In the discrete case, the tabu mechanism relies on an equality test on the configurations. This cannot transpose to interval domains, where two intervals have a near to zero chance to be equal. A tabu configuration must forbid the search not only at a point, but in an area around it. A simple way to ensure this is to compare the configurations not w.r.t. equality, but w.r.t. intersection We add the possibility to resize the tabu configurations, in order to control their tabu influence. This `resize` function multiplies all lengths by a constant factor α. Intuitively, a configuration will be left out of the search if it crosses more than α of a tabu configuration. In the end, the tabu mechanism is enclosed into a single function $\texttt{cutTabu}(\mathcal{T}, V) = \{v \in V, \forall v_{\text{tabu}} \in \mathcal{T}, v \cap \texttt{resize}(v_{\text{tabu}})) = \emptyset\}$

2.2 Implementation

We have instantiated our framework to compute inner-approximations for two non-linear CSPs: continuous NQueens and Nlights[1]. Both have non-smooth con-

[1] Available online with implementation and results at `http://www.normalesup.org/~truchet/LSCont/`

nb_tries $\leftarrow 0$, $\mathcal{T} \leftarrow \emptyset$
repeat
  iter $\leftarrow 0$, $s \leftarrow$ `random`$(\mathcal{S})$
  repeat
    $N_0 \leftarrow$ `neighborhood`(s), $N_1 \leftarrow$ `sample`(N_0) // neighbourhood generation
    $N_2 \leftarrow$ `cutTabu`$(N_1, \mathcal{T})$ // suppression of tabu configurations
    $s \leftarrow$usualRandom$\{s' \in N_2, f(s')$ is minimal $\}$ // selection of a best neighbour
    $\mathcal{T} \leftarrow (s, \mathtt{t}) \cup$ `actualizeTabu`$(\mathcal{T})$, iter++
  until iter¿ max_iter **or** $f(s) = 0$
  nb_tries++
until nb_tries ¿ max_tries **or** $f(s) = 0$

Fig. 1. Continuous Tabu Search

straints, which impedes the use of classic derivative techniques. Neighborhood and sampling have been defined and tested with different strategies. The implementation enables to prove the expressiveness of the framework.

Table 1. Results over the Interval implementation of the Tabu Search Framework (average on 20 runs)

Benchmark	NbSteps max_iter	NbNeighbors nb_{neigh}	Tabu Tenure t	%Success	Nbrestarts max_tries=30	Time
NQueens 5	100	30	5	100%	1.0	0.08 s.
NQueens 8	500	50	5	100%	1.0	2.2 s.
NQueens 10	500	100	5	100%	1.5	5.07 s.
NLights 10	100	30	5	100%	2	0.95 s.
NLights 10	100	50	5	100%	1	0.61 s.
NLights 15	500	100	5	100%	1	7.56 s.

In terms of performances, a comparison is difficult to establish as most techniques do not compute inner-approximations of systems, and those who do, rely on a complete exploration of the search space (*e.g.* Realpaver). However, the system was able to solve instances with many variables on which Realpaver fails in a reasonnable time (10 min). Table 1 shows some results for different parameter settings.

2.3 Discussion

As a first remark, the framework allows to retrieve other algorithms (or part if them if hybridized), for instance ECTS [2] by implemeting its distance function in `neighborhood` and `resize`.

As often, the algorithm has some critical parameters. In addition to the usual TS ones, we introduce two parameters: nb_{neigh} and α. This is of course a drawback. However, nb_{neigh} can be tuned rather easily, by counting the time spent in neighborhood exploration w.r.t. the time spent in a single iteration. And α behaves in the same way as `t`.

The main advantage of the proposed framework is its genericity. All the key ingredients, which define the search strategy, are defined independantly of the CSP. Constraints can also be generic, although, truth be told, the TS algorithm is unlikely to challenge problems with well known constraint types (linear, polynomial, differentiable functions). But it offers to solve in an unified way constraints that could not be properly handled by classical approaches.

3 Conclusion

The proposed framework extends the TS metaheuristic to handle interval CSPs and their particular semantic in a generic way. Further work has to be done to try other possibilities for the new TS operators. The framework can also be extended to other kind of set CSPs, provided they come with a finite representation and computable constraints.

Some continuous problems are still a challenge for resolution methods, and the interval TS is an good alternative in that case. Experimentally, we provide two such CSPs. The interval TS works well on them because of their high number of variables and constraints, with high correlation within the variables in the constraints.

References

1. Barichard, V., Hao, J.-K.: A Population and Interval Constraint Propagation Algorithm. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 88–101. Springer, Heidelberg (2003)
2. Chelouah, R., Siarry, P.: Tabu search applied to global optimization. European Journal on Operational Research (2000)
3. Cruz, J.: Constraint Reasoning for Differential Models. IOS Press, Amsterdam (2005)
4. Fanni, A., Manunza, A., Marchesi, M., Pilo, F.: Tabu search metaheuristics for electromagnetic problemsoptimization in continuous domains. IEEE Transactions on Magnetics 35(3) (1999)
5. Franze, F., Speciale, N.: A tabu-search-based algorithm for continuous multiminima problems. International Journal for Numerical Methods in Engineering 50(3) (2001)
6. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht, The Netherlands (1997)
7. Granvilliers, L., FrédéricBenhamou: Algorithm 852: RealPaver: An interval solver using constraint satisfaction techniques. ACM Transactions on Mathematical Software 32(1) (2006)
8. Hansen, E.R.: Global Optimization Using Interval Analysis. Pure and Applied Mathematics. Marcel Dekker Inc, New York (1992)
9. Moore, R.E.: Interval Analysis. Prentice-Hall, Englewood Cliffs (1966)