Transition Models as an incremental approach for problem solving in Evolutionary Algorithms

Anne Defaweux
Vrije Universiteit Brussel
Faculty of Science,
DINF-COMO
Pleinlaan, 2
Brussels, Belgium
adefaweu@vub.ac.be

Tom Lenaerts
Université Libre de Bruxelles
Faculty of Applied Science,
IRIDIA
50, Av. F. Roosevelt, CP 194/6
Brussels, Belgium
tlenaert@ulb.ac.be

Jano van Hemert
Napier University
Centre for Emergent
Computing
10 Collinton Road
Edinburgh, United Kingdom
j.van.hemert@napier.ac.uk

ABSTRACT

This paper proposes an incremental approach for building solutions using evolutionary computation. It presents a simple evolutionary model called a Transition model in which partial solutions are constructed that interact to provide larger solutions. An evolutionary process is used to merge these partial solutions into a full solution for the problem at hand. The paper provides a preliminary study on the evolutionary dynamics of this model as well as an empirical comparison with other evolutionary techniques on binary constraint satisfaction.

Categories and Subject Descriptors

Track [**Evolutionary Combinatorial Optimization**]

General Terms

Algorithms

Keywords

Evolutionary Algorithms, Constraint Satisfaction Problem, Emergence of Complexity, Local search, Combinatorial Optimization

1. INTRODUCTION

In the approach of stochastic local search methods such as Evolutionary Algorithms (EAs), solutions are represented as fully defined solutions to the problem [7]. These fully qualifying representations are evolved toward an optimum by means of optimisation techniques that aim to traverse the search space in the most efficient and adequate way to the problem at hand. In this paper, we are interested in developing an evolutionary algorithm that does not evolve fully qualifying solutions but rather builds such fully defined solutions as collaboration of very simple building units. In other words, collaboration provides an incremental mechanism to produce more complex solutions. We believe this approach is promising on problems in which a structure in the components of the solution exists but for which this structure is unknown before hand and therefore needs to be learned during the optimisation process. This approach is related to the principal of divide-and-conquer. Yet no division step occurs. The incremental approach we use to build solutions is referred to as a Transition model, as a reference to the biological counterpart that inspired this work [16, 12].

To illustrate the Transition model, we will discuss it within the context of Binary Constraint Satisfaction Problems (BINCSP). BINCSP forms an interesting class of problems to work on. First, it is an already well studied problem class [17, 1] and therefore, it can be used as a firm benchmark for comparison with other evolutionary techniques [19, 4]. Second, it is not a toy problem specifically made for illustrating our model, but is a NP-complete problem where we shall deliberately use hard to solve problem instances to evaluate our proposed Transition model. Third, and most importantly, BINCSP instances can easily be described in terms of the aggregations of simple solution units. The idea of aggregating simple solution units was used before to speed up constraint programming techniques that learn which aggregations are undesirable and, hence, should be avoided [6]. Here we shall take the opposite approach whereby we evolve aggregations which contribute to solving the problem, i.e., are desirable.

The structure of the paper is the following: We first introduce related work on the topic of incremental search. We then pursue with a short introduction to Binary Satisfaction Problem and how our Transition model can solve such a problem. We explain the outcome of some experiments that validate our model and finally we conclude.

2. RELATED WORK

The first attempts to introduce collaboration and cooperation in evolutionary computation were strongly related to a divide and conquer approach where sub-problems are derived from the entire problem. Each sub-problem is solved through evolutionary techniques and the collaboration of the solutions yields a complete solution for the problem at hand [8, 13]. The divide and conquer approach however is the result of engineering techniques and lacks the principle of emergence of the complete solutions from the interactions of simple ones.

Other grouping techniques exist too. Among them, an interesting approach is given by the so called Multi-Level Selection Models [9, 10]. In this model, solutions are spread into groups and evolve within these groups. This approach intents to favour the specialisation of solutions into solving certain aspects of the problem. Although it evolves good collaborating units, it does not address the topic of incremental search.

Finally, the symbiogenetic model proposed by Watson [20, 21] is one of the few mechanisms which tries to address this problem of incremental search. This approach suggests to consider incompletely defined solutions. The undefined part of the solutions are filled in with "don't care" symbols. In order to evaluate a solution, a fully defined solution is obtained by aggregating the solutions with others until a full description of a solution is obtained. This way, solutions evolve within a context defined by the other solutions and solutions that solve nicely a part of the problem are more likely to be selected for the next generation. There is however a serious cost in this approach. In order to evaluate a solution, several contexts need to be built, this yields a serious overhead in the evaluation process. Furthermore, this approach still requires fully described solutions for the evolutionary process to perform, among others, evaluation and selection.

3. BINARY CONSTRAINTS SATISFACTION PROBLEMS

Constraint Satisfaction Problems (CSP) [17] form a problem class, which is NP-complete, where we have on the one hand a set of variables X associated with possible domain values D and on the other hand a set of constraints C defined on this set of variables, which prohibits combinations of assignments to occur. The problem consists of finding an assignment to the whole set of variables from the associated domain values so that all constraints are satisfied. If this proves to be impossible then the corresponding problem is said to be unsolvable. A variant of this problem is the BINCSP, where each constraint is defined on at most two variables. This forms no restriction on the general form of CSP as every CSP can be rewritten into a BINCSP and vice versa [14].

Let us take as an illustration the following BINCSP: consider a set of six variables: $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ all taking values in $D = \{1, 2, 3\}$. We consider the following set of constraints:

$$\begin{aligned} C = \quad & \{(x_1 \neq x_2), (x_2 \neq x_3), (x_3 \neq x_1), \\ & (x_4 \neq x_5), (x_5 \neq x_6), (x_6 \neq x_4), \\ & (x_1 = x_4), (x_2 = x_5), (x_3 = x_6)\} \end{aligned} \tag{1}$$

This setup of constraints consists of nine binary constraints. Each binary constraint defines a relation on two variables. Thus, for each pair of variables, only one binary constraint may be defined.

The problem consists of finding a correct assignment for the variables, one which satisfies all the constraints. We denote the assignment of one variable $x_i \in X$ with value $d \in D$ by $\langle d, i\rangle$ where i is the index of the variable we consider. Using this notation, we represent the simultaneous assignments of variables x_1, x_2 and x_4 with respective values v_1, v_2 and v_4 as

$$(\langle v_1, 1\rangle, \langle v_2, 2\rangle, \langle v_4, 4\rangle)$$

4. TRANSITION MODEL FOR PROBLEM SOLVING

In this section, we will illustrate the key features of our Transition model by using a simple example of a binary constraint satisfaction problem, described by equation (1).

Before going into the example, we introduce the reader with a number of definitions that we use in the description of the model:

- A *Partial Solution* is the assignments of a proper subset of the variable set. An example of a partial solution will be given by (2).
- A *Solution* is the case of a partial solution where assignments are defined for the entire variable set.
- A *Genotype* is the representation of the simultaneous assignment of a partial solution. For example, (2) is the genotype of the solution that assigns variables 1 and 2 with respective values 1 and 3.
- A *Symbiotic Partner* is another (partial) solution with which a solution shall cooperate. (An example is given by (3)).
- A *(induced) phenotype* of a solution is the variables assignment one obtains when working out the symbiotic relation of the solution with its symbiotic partner. The way such a phenotype is obtained and its notation is explained in the next section.

4.1 Basic Representation

Our model is a simple generational evolutionary algorithm that starts with a population of partially defined solutions. A partial solution s that only defines values for x_1 and x_2:

$$s = (\langle 1, 1\rangle, \langle 3, 2\rangle). \tag{2}$$

(2) is called the genotype of the solution. In our example problem, a solution is said to be fully qualifying when assignments are defined for all six variables.

The basic principle of incremental search is to place these partially defined solutions in an interaction framework. For example, let solution (2) interact with

$$sp = (\langle 3, 1\rangle, \langle 2, 3\rangle).$$

We call this interacting partner[1] the symbiotic partner sp of (2) as reference to the biological counterpart of our model and denote the relation by:

$$(\langle 1, 1\rangle, \langle 3, 2\rangle) \leftrightarrow (\langle 3, 1\rangle, \langle 2, 3\rangle). \tag{3}$$

The underlying idea behind interaction is information sharing between cooperative individuals, i.e., solutions. The way information is shared between the two solution described in (3) is the following:

- The solution genotype is extended with the assignments found in its symbiotic partner:

$$\langle \begin{pmatrix} 1 \\ 3 \end{pmatrix}, 1\rangle, \langle 3, 2\rangle, \langle 2, 3\rangle \tag{4}$$

[1]At this time, we limit ourselves to pairwise interaction, however, interaction could also happen with more than one partner and will be studied in further work

- Conflicting values are solved by randomly selecting one of the two values with equal probability. In our example, a conflict needs to be solved for variable x_1. We can choose between values 1 and 3. A possible conflict resolution for this example would be:

  $$\phi(s, sp) = (\langle 1, 1\rangle, \langle 3, 2\rangle, \langle 2, 3\rangle) \tag{5}$$

  where $\phi(s, sp)$ defines the induced phenotype between s and sp. This phenotype is used for evaluating the solution and the result of the evaluation is assigned to (2). We use random pairwise interactions of solutions in our current implementation of the model.

4.2 Evaluation

In the context of BINCSP, we will consider two types of evaluation. The first type considers the quality of the partially defined solution relative to the entire constraints set. This evaluation function corresponds to the classical approach of fitness computation for CSP solving by EAs [11]. That is, it is defined as the ratio of constraints from the constraints set that is satisfied by the solution. The second type restricts itself to the subset of constraints that are covered by the partial solution.

Let us denote by $c_k(p)$ the outcome of evaluating phenotype p with constraints k, we say that p covers c_k if p contains assignments for all variables contained in c_k, furthermore, p satisfies c_k if the assignment values in p match the constraints defined by c_k.

$$c_k(p) = \begin{cases} 1 & p \text{ covers } c_k \ \wedge p \text{ satisfies } c_k \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Given this, the classical evaluation of the solution described by (2) and denoted by s working with a symbiotic partner sp is given by:

$$f(s) = \frac{1}{|C|} \sum_{k \in C} c_k(\phi(s, sp)) \tag{7}$$

where C is the set of pairs of conflicting values, $|C|$ the size of this set and $\phi(s, sp)$ the induced phenotype of s when sharing information with its symbiotic partner.

This fitness value is a measure of the quality of the partially defined solution with respect to the entire set of constraints. It does not, however, give any indication of the quality of the partially defined assignments with respect to the constraints it covers. To see whether an association is beneficial or not, we define a restricted fitness measure that only considers the constraints covered by the phenotype of the solution. If we denote by $covers(s, C)$ the set of constraints covered by s, the covering fitness measure is given by:

$$f_{cover}(s) = \frac{1}{covers(\phi(s, sp), C)} \sum_{k \in covers(\phi(s,sp),C)} c_k(\phi(s, sp)) \tag{8}$$

We use the first measure (7) to guide the evolutionary process (selection). The second measure is used to decide whether a solution and its symbiotic partner should be merged into a more qualifying solution through the use of a transition. Transitions are described next.

4.3 Reproduction and Transition

In our evolutionary process, solutions are selected according to their fitness described by (7). We consider a very simple evolutionary process that consists of replicating, i.e., copying highly fit individuals where small probability of mutation is included. When a solution is selected, it will therefore be replicated into a new solution.

In it most simple form, the replication looses the symbiotic link that binds the parents, i.e., the interaction does not survive more than one generation. However, a solution may help its symbiotic partner to replicate. Currently, this replication of the symbiotic partner is based on a random selection, but it could also be based on more problem specific strategies. When the symbiotic partner is replicated as well, the symbiotic link, that is, their interaction scheme will be inherited in the process. The underlying idea is that (possibly) good collaborating units can survive over more than one generation.

The replication process as it is described up to now, does not allow us to evolve more complex solutions. To perform this task, partial solutions, selected based on (7), are evaluated with (8). If this covering fitness is larger than a certain threshold value, the solution will be allowed to replicate its induced phenotype. In this paper, we set this threshold value equal to 1.0 which means that we request the symbiotic relation to solve perfectly the sub-problem at hand. The goal of this transitional step is to fixate the collaboration that succeeds in solving the sub-problem defined by the covering set of constraints induced by the phenotype. This new (partial) solution represents a new entity defined at a higher level which can interact with other solutions and produce new symbiotic relations. The general idea behind our approach is to allow the emergence of useful entities at higher levels only [2].

Let us take the example solution discussed previously. The solution described by (2) with the phenotype given in (5) had a classical fitness value of 0.33. The measure of the fitness restricted to the covering constraints set was 1. In this case, if the solution is selected, the phenotype and not the genotype will be replicated and passed on to the offspring. This means that the representation of the offspring genotype will be: $(\langle 1, 1\rangle, \langle 3, 2\rangle, \langle 2, 3\rangle)$ instead of $(\langle 1, 1\rangle, \langle 3, 2\rangle)$. This is the point of our incremental approach. Solution are grown incrementally based on their success in solving the sub-problem they define within the whole problem.

4.4 A word on recombination

This evolutionary algorithm has no explicit recombination operation. However, it does combine sub-solutions through symbiotic relations. This recombination is far from the general EA approach, which starts from a genotype that represents a complete candidate solution to the whole problem and recombines these genotypes to create new candidate solutions. In our approach, the first phase consists of combining entities of the genotype in order to build fully grown genotypes. However, once the genotypes are becoming large, there is a conflict mediation process which recombines the conflicting part of the genotypes, i.e., chooses which conflicting values need to be used in the resulting phenotype. This conflict mediation is comparable with a uniform crossover operation in the special case when two fully qualifying genotypes are interacting.

So, even if we cannot talk of a recombination operator because of the way the Transition model evolves basic entities into complex ones, there are however similarities between the recombination operators of EAs and the conflict mediation operation of the Transition model on one hand and between the building block theory from EA and the way genotypes are built in the Transition model on the other hand.

5. EXPERIMENTATION

5.1 Goal

The first goal of our experimentation is to observe whether the model we made succeeds in incrementally building fully defined solutions from smaller solution units. If more complex can emerge out of simple units, this means that the process succeed in aggregating the correct units. By complexity we mean solutions that are more and more precisely defined for the entire problem.

Furthermore, we would like to see how this model behaves relative to other evolutionary techniques. We compare the results of these experiments with three other evolutionary techniques[5]:

- The co-evolutionary Constraints Satisfaction (CCS): In this algorithm, the set of constraints is considered as an interacting population with the solution population. This mimics an arms-race principle where the constraints used for evaluating the solutions are selected amongst those which score best, that is for which solutions score badly. In the same way, the set of solutions to test the constraints are chosen amongst those which score best.

- The Micro-genetic algorithm with iterative descent (MID): This algorithm introduces heuristics to help the search process. The first heuristic is a specialised mutation operator that operates like a hill-climbing algorithm on one variable at a time. The second heuristic consists of including a *Breakout Management Mechanism* to escape local optimum that adapts weights in the evaluation function in such a way that the violated constraints become more coercitive on the final result and therefore encouraging the solution to move towards another peak.

- The Stepwise Adaptation of Weights (SAW): The underlying idea of this algorithms is to let a vector of weights used in the evaluation function evolve with the solution populations. This way, it forces the evolutionary algorithm to focus itself on optimising different part of the problem. These weights are increased over time for the parts of the problem which are considered as badly solved by the best individuals.

5.2 Experimental setup

We use randomly generated problem instances of BINCSP for benchmarking the Transition model against the three evolutionary algorithms described above. The RandomCSP package [18] is used to generate the suite of test problem instances [19]. To scale the difficulty of the problem instances, these BINCSPs are generated accordingly to two parameters. The first parameter is the density of the constraint graph of the BINCSP: If we consider C as the set of constraints and n as the number of variables in the BINCSP,

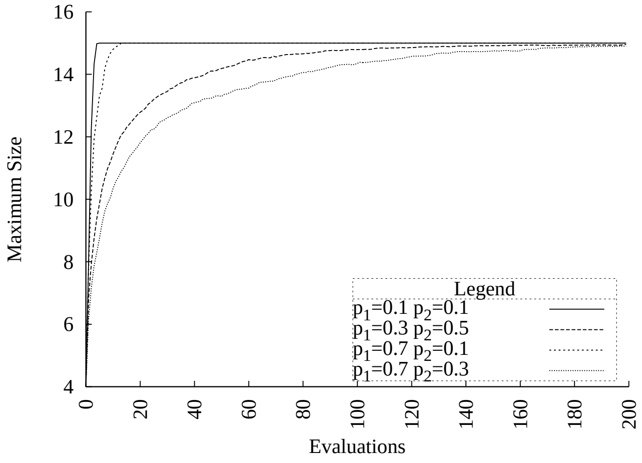

Figure 1: Evolution of the genotype sizes for easy BINCSP for the Transition Model

we have,

$$p_1 = \frac{|C|}{\binom{n}{2}}.$$

This parameter reflects how many constraints there are relative to the maximum amount of constraints possible.

The second parameter is the average tightness $\bar{p}_2$ of constraints, which reflects the average complexity of the constraints. That is, how many invalid simultaneous assignments of two variables are allowed given one constraints, averaged over all constraints. If we denote by $|c|$ the number of assignments in the constraints c resulting in a non violation of this constraints, and by m the domain of the variables in constraints c, we get,

$$\bar{p}_2 = \frac{1}{|C|}\sum_{c \in C}\left(1 - \frac{|c|}{m^2}\right)$$

p_1 and $\bar{p}_2$ are so called order parameters, as they can be used to order the problem instances of a class of problems. By fixing the number of variables and the domain size of each variable, and then varying these parameters we can induce a phase transition [3]. For low values of the parameters, all problem instances will be solvable. When increasing them, at some point, unsolvable problems appear, and at some higher values, all the problems become unsolvable. The conjecture is that the location of the phase transition as given by parameter coordinates coincides with where the hardest to solve (or to proof unsolvable) problems occur. This gives rise to the typical easy-hard-easy pattern.

5.2.1 Simulation Parameters

We vary p_1 and $\bar{p}_2$ from 0.1 up to 0.9 with a step size of 0.2. For each combination of p_1 and $\bar{p}_2$, we generate 25 random problem instances for a BINCSP of 15 variables each taking values in a domain of size 15.

For each of the 25 problem instances, we perform 10 runs (each run using a different seed, a population size of 100 and a mutation rate of 0.001). The maximum amount of evaluations is set to 100 000.

5.2.2 Observations

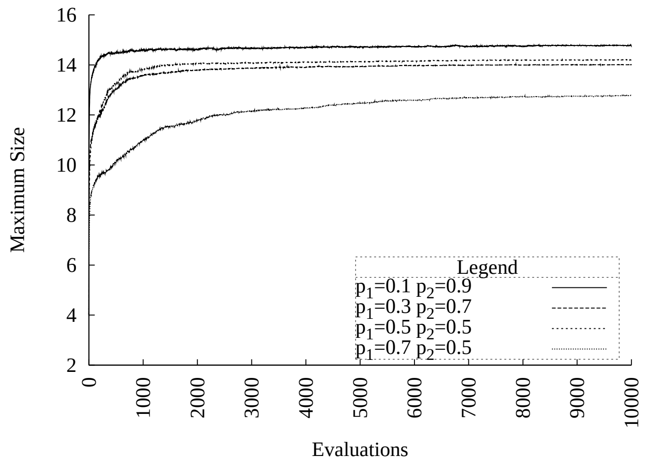

Figure 2: Evolution of the genotype sizes for difficult BINCSP for the Transition Model

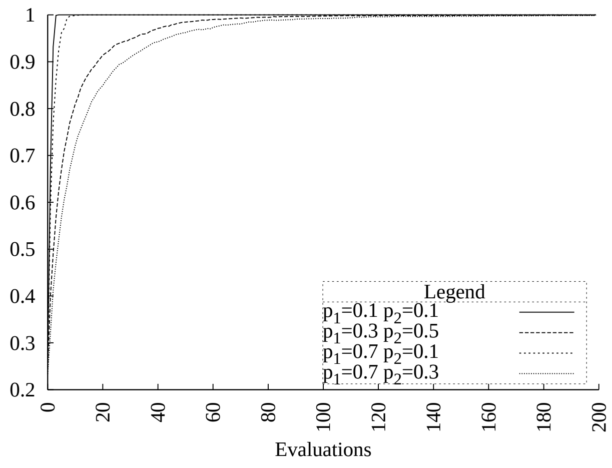

Figure 3: Evolution of the best fitness for easy BINCSP for the Transition Model

We collected two types of data over all the runs. The first set of data should help us to determine whether the Transition model succeeds in building incrementally more complex solution instances. The second set of data is used to compare our algorithm with the three others. We are interested in following information:

- Solution sizes: We will observe what is the maximum and minimum size of the solutions over time, together with the amount of overlap between the symbiotic partners.
- Fitness: We are interested in the fitness evolution over time, the best achieved fitness, the average fitness of the population, the fitness of the biggest individual.
- Success ratio: This gives the ratio of successful runs over the total amount of runs
- Average amount Evaluations to reach a solution: Since each generation requests exactly one evaluation of the population, this number is strongly correlated to the number of generations and therefore gives an idea of the speed of the algorithm to find a solution. Also, the fitness calculation forms the most computational expensive component.

5.3 Results

5.3.1 *Examining Incremental Approach Dynamics*

The evolution of the average genotype sizes for easy and difficult BINCSP can be seen in Figure 1 and 2. We observe that for easy BINCSP, the Transition model succeeds in finding the solution (in this case, a genotype of size 15) in at most 200 generations. Also, we see from table 1 that, for the easy BINCSPs, the speed of the algorithm to find a solution is correlated to the parameter p_1 of the BINCSP. This means that it is closely related to the complexity of the constraints network. For a small amount of constraints, the connectivity is not too high and the problem can be seen as a collection of sub-problems. For these problems, the sub-parts of the problems are easily solvable by the Transition model. Partial solutions for these sub-parts are soon evolved and by combining these partial solution, one arrives easily at a solution for the entire problem.

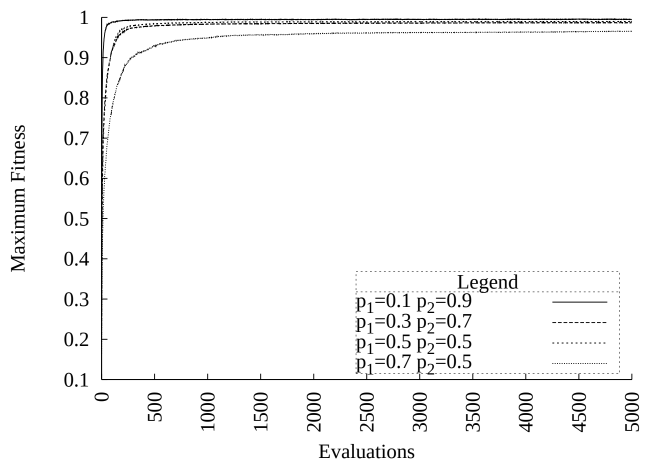

Figure 4: Evolution of the best fitness for difficult BINCSP for the Transition Model

In Figure 2, the constraints prove to be more difficult to learn. For low values of p_1, a complete solution emerges almost every time. However, as the constraints become more difficult to learn, i.e., higher values in the combination of p_1 and p_2, the evolutionary process fails to build fully defined solution. The partially defined solutions cooperate in order to find a best possible compromise to the problem but fails in solving the conflicts in some of their variables. To observe that the evolved genotypes are indeed evolving toward a good compromise, we look at the evolution of fitness over time.

Figure 3 and 4 shows the evolution of average best fitness for the first generations for easy and difficult BINCSP, respectively. The first graph show without doubt that the Transition model succeeds in finding a solution as the fitness converges rapidly to 1.0. The second graph shows that over 5 000 generations, the algorithms converges to a combination of partial solutions that solves over 90% of the constraints.

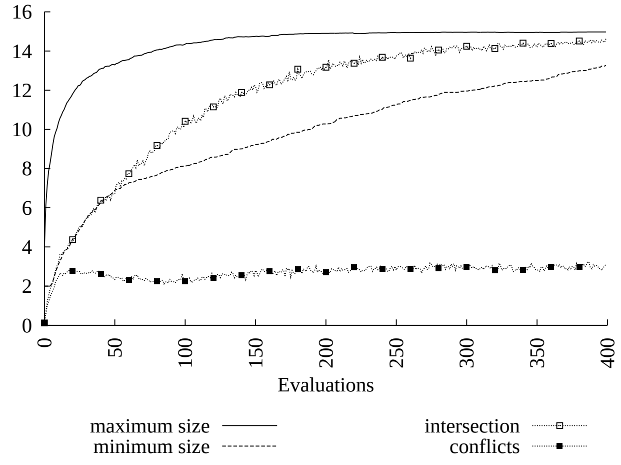

Figure 5: Variation of the genotype size (max, min and the size of the common variable of best individual with its symbiotic partner as well as the amount of conflicting variable amongst this intersection for the easy BINCSP $p_1 = 0.1$ and $p_2 = 0.1$

Both of these graphs illustrate the remarks we made previously concerning the sizes of the genotype. The Transition model performs better on constraint graphs which are more likely to show structure than on ones that are highly unstructured. The curve concerning the most difficult problems to solve, i.e., $p_1 = 0.7$ and $\bar{p}_2 = 0.5$, confirms the idea that genotypes are evolving into two interacting genotypes which together are able to solve over 90% of the constraint graphs but fail in solving the remaining conflicts, which therefore prevent them from finding a solution to the whole problem.

5.3.2 Evolution of the genotype complexity over time

By genotype complexity, we mean the length a genotype attains over time as the transitions occur. In Figure 5 and 6, we plot the evolution of the maximum size and minimum size of the genotype in the population. Next to these two values, we plot the evolution of the size of the intersection of the best solution genotype with its symbiotic partner's genotype. This intersection gives a measure of the complementarity of the solution with its symbiotic partner. Finally, we also plot the evolution of the number of conflicting variables in this intersection. This measurement of conflicts gives an idea of the variation present in the population. A high value for the intersection correspond to a low complementarity. A low value of conflicts combined with a big intersection results in a algorithm which has converged to a solution.

In Figure 5, we observe that for easy BINCSP, the intersection is small while exploring, which corresponds to a high complementarity. Once, the algorithm begins to converge towards a solution, this complementarity decreases. However, the measurement of the conflicts in the intersection increases. The measurement of the conflicts in this intersection reflects however a low but existing variation in the population genotype throughout the evolutionary process.

In Figure 6, We show that for difficult BINCSP, a quite similar evolution occurs. However, in the such a case, the evolved genotype needs a symbiotic partner to cover the

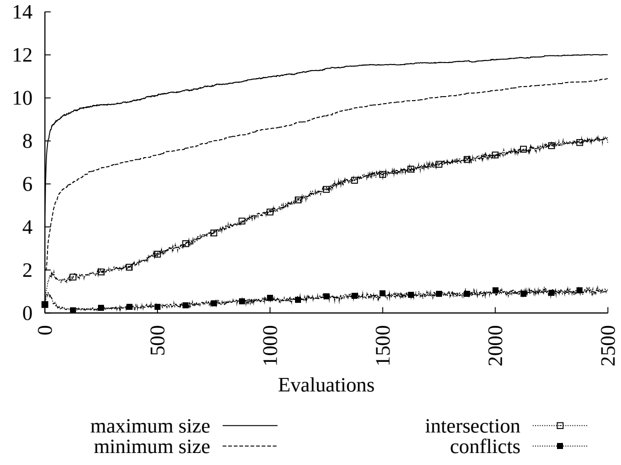

Figure 6: Variation of the genotype size (max, min and the size of the common variable of best individual with its symbiotic partner as well as the amount of conflicting variable amongst this intersection for the difficult BINCSP $p_1 = 0.7$ and $p_2 = 0.5$

Density	Tightness				
p_1	0.1	0.3	0.5	0.7	0.9
0.1	1.0 (2.6)	1.0 (3.7)	1.0 (6.5)	1.0 (14.8)	0.98 (3352.9)
0.3	1.0 (3.5)	1.0 (12.8)	1.0 (61.1)	0.21 (7363)	- -
0.5	1.0 (4.9)	1.0 (32)	0.51 (14426)	- -	- -
0.7	1.0 (7.5)	1.0 (89.8)	0.31 (8710)	- -	- -
0.9	1.0 (10.6)	0.98 (3961)	- -	- -	- -

Table 1: success ratio and Average Number of Evaluation for the Transition model

entire variable set which implies that the amount of complementarity is sustained throughout the evolutionary process. On the other hand, The low value in the measurement of conflicts reflects the nature of this algorithms to simultaneously evolve two complementary partial solution that are specialised in solving two different sub-sets of the constraints set. However, the difficulty of solving the problem makes it impossible to solve certain conflicts amongst these variables thereby making it impossible to evolve a fully defined solution.

5.3.3 Evaluating Incremental Approach with other Techniques

The results of the four algorithms for varying values of p_1 and $\bar{p}_2$ are given in table 1 to 4. In each table, we show the success ratio obtained together with the average number of necessary evaluations to get a solution for each setup of p_1 and $\bar{p}_2$.

From these tables, we see that the Transition model outperform CCS on all matters. However, comparing results with MID or SAW does not show improvements for the Transition model, on the contrary, it is more likely that MID and

Density p_1	Tightness 0.1	0.3	0.5	0.7	0.9
0.1	1.0 (3)	1.0 (15)	1.0 (449)	1.0 (2789)	0.62 (30852)
0.3	1.0 (96)	1.0 (11778)	0.18 (43217)	0.0 (-)	- -
0.5	1.0 (1547)	0.08 (39679)	0.0 (-)	- -	- -
0.7	1.0 (9056)	0.0 (-)	0.0 (-)	- -	- -
0.9	0.91 (28427)	0.0 (-)	- -	- -	- -

Table 2: success ratio and Average Number of Evaluation for CCS

Density p_1	Tightness 0.1	0.3	0.5	0.7	0.9
0.1	1.0 (1)	1.0 (4)	1.0 (21)	1.0 (87)	0.96 (2923)
0.3	1.0 (3)	1.0 (50)	1.0 (323)	0.52 (32412)	- -
0.5	1.0 (10)	1.0 (177)	0.90 (26792)	- -	- -
0.7	1.0 (20)	1.0 (604)	0.0 (-)	- -	- -
0.9	1.0 (33)	1.0 (8136)	- -	- -	- -

Table 3: success ratio and Average Number of Evaluation for MID

Density p_1	Tightness 0.1	0.3	0.5	0.7	0.9
0.1	1.0 (1)	1.0 (1)	1.0 (2)	1.0 (9)	0.64 (1159)
0.3	1.0 (1)	1.0 (2)	1.0 (36)	0.23 (21281)	- -
0.5	1.0 (1)	1.0 (8)	0.74 (10722)	- -	- -
0.7	1.0 (1)	1.0 (73)	0.0 (-)	- -	- -
0.9	1.0 (1)	1.0 (3848)	- -	- -	- -

Table 4: success ratio and Average Number of Evaluation for SAW

SAW better perform on harder instances than the Transition model. In general, the performance of the Transition model is comparable to those of SAW, and it uses less evaluations to find a solution than MID. However, it does not find a solution as often as MID does.

One exception, however, concerns the case where p_1 is low (0.1) and $\bar{p}_2$ is high (0.9). Here, the Transition model finds more often a solution than the other three algorithms. This case yields a possible explanation for the reason Transition models do not perform well on the other hard instances. The incremental approach consists of letting solution growing as they succeed in solving sub-parts of the problem instance. If the problem instance contains some structure in the network of constraints, the Transition model is more likely to learn this structure. This is a consequence of the incremental nature of our Transition model. When there is no structure, i.e., a very rugged landscape with high epistasis (which can be directly correlated to higher values of p_1), the Transition model will get stuck in partially interacting solution. Good compromising partial solutions may still form, but no fully satisfying solutions will be found.

The test set of random BINCSPs we used are not built to show specific structure, i.e., to be decomposable into interdependent sub-problems. However, for a low value of p_1, the network of constraints is not very loaded. So, even if the constraints to learn are difficult (high $\bar{p}2$), the Transition model will conquer these constraints iteratively by solving them up separately and aggregating them together. The Transition model is therefore more likely to succeed than a global approach like SAW or CCS. MID does perform almost as well as the Transition model on this problem by the nature of its greedy search on one variable at a time. However, for this special case, we can conclude that the Transition model shows improvement comparatively to other co-evolutionary techniques.

One remark still needs to be stressed. It concerns the average number of evaluations to a solution for the special case $p_1 = 0.1$ and $\bar{p}_2 = 0.9$. We observe that the number of evaluations to a solution is relatively larger for the Transition model compared to SAW and MID. This yields to the conclusion that the improved performance in finding a solution is compensated with the necessary time to find such a solution.

6. CONCLUSIONS

In this paper, we propose an alternative approach for problem solving in evolutionary computation that consists of incrementally building new solutions by combining evolving solution units. We call this incremental model a Transition model by reference to the biological term it is inspired from. To test our model, we apply it to binary constraint satisfaction, where it is tested on a range of problems scaling from easy to solve up to difficult to solve.

The results from our experiments show that our Transition model performs almost as well as specifically designed evolutionary techniques for this problem. Furthermore, we can deduce from these experiments that the Transition model is more likely to perform better on problem instances that induce some form of structure in the search space when being solved. However, this structure needs not to be known beforehand as the Transition model successfully learns it as the evolutionary process goes along.

We also show that the interaction schema that rules the partial solution of our model succeeds in combining these to a full solution or at least a good compromised solution. This interaction schema, together with the transition feature of our model, achieves the evolution towards complex solutions from basic solution units as the result of an emergent process.

Acknowledgements

The third author is supported through a TALENT-Stipendium awarded by the Netherlands Organization for Scientific Research (NWO).

7. ADDITIONAL AUTHORS

Additional authors: Johan Parent (Vrije Universiteit Brussel, Faculty of Applied Science, ETRO, Pleinlaan, 2, Brussels, Belgium. Email: `jparent@info.vub.ac.be`)

8. REFERENCES

[1] D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, and Y. Stamatiou. Random constraint satisfaction: A more accurate picture. *Constraints*, 4(6):329–344, 2001.

[2] H. Bersini. Whatever emerges should be intrinsically useful. In *Proceedings of the ninth international conference on artificial life*, pages 226–231. MIT press, 2004.

[3] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*, 1991.

[4] B. Craenen, A. Eiben, and J. van Hemert. Comparing evolutionary algorithms on binary constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 7(5):424–444, 2003.

[5] A. Eiben, J. van Hemert, E. Marchiori, and A. Steenbeek. Solving binary constraint satisfaction problems using evolutionary algorithms with an adaptive fitness function. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, number 1498 in Springer Lecture Notes on Computer Science, pages 196–205. Springer-Verlag, Berlin, 1998.

[6] D. Frost and R. Dechter. Dead-end driven learning. In *Proceedings of the Twelfth National Conference of Artificial Intelligence (AAAI-94)*, volume 1, pages 294–300. AAAI Press, 1994.

[7] H. H. Hoos and T. Stützle. *Stochastic Local Search, Foundation and Applications*. Morgan Kaufmann Publishers, 2005.

[8] K. A. D. Jong and M. A. Potter. Evolving complex structures via cooperative coevolution. In *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*. MIT press, 1995.

[9] T. Lenaerts, A. Defaweux, P. van Remortel, and B. Manderick. An individual-based approach to multi-level selection. In *the Genetic and Evolutionary Computation Conference*, pages 136–144. Morgan-Kauffman, 2002.

[10] T. Lenaerts, A. Defaweux, P. van Remortel, and B. Manderick. Modelling artificial multi-level selection. In *AAAI Spring Symposium on Computational Synthesis*. AAAI Spring Symposium Series, Stanford, USA, 2003.

[11] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In J. McDonnell and R. Reynolds, editors, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 135–155. The MIT Press, 1995.

[12] R. Michod. *Darwinian Dynamics: Evolutionary transitions in Fitness and Individuality*. Princeton University Press, 1999.

[13] M. Potter. *The Design and Analysis of a Computational Model of Cooperative Co-evolution*. PhD thesis, George Manson University, Department of Comuter Science, 1997.

[14] F. Rossi and V. Dhar. On the equivalence of constraint satisfaction problems. In L. C. Aiello, editor, *ECAI'90: Proceedings of the 9th European Conference on Artificial Intelligence*, pages 550–556, Stockholm, 1990. Pitman.

[15] S. Russel and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice-Hall Series in Artificial Intelligence, 1995.

[16] E. Sober and D. Wilson. *Unto Others, the Evolution and Psychology of Unselfish Behaviour*. Cambridge, MA:Harvard University Press, 1998.

[17] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press Limited, 1993.

[18] J. van Hemert. RandomCSP. Freely available from `http://freshmeat.net/projects/randomcsp/`.

[19] J. van Hemert. *Application of Evolutionary Computation to Constraints Satisfaction and Data Mining*. PhD thesis, Universiteit Leiden, 2002.

[20] R. A. Watson and J. B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In *Parallel Problem Solving from Nature (PPSNVI)*. Springer Verlag, Lecture Notes in Computer Science 1917, 2000.

[21] R. A. Watson and J. B. Pollack. A computational model of symbiotic composition in evolutionary transitions. *Biosystems Special Issue on Evolvability*, 69/2-3:187–209, 2002.