Complexity Transitions in Evolutionary Algorithms: Evaluating the impact of the initial population

Anne Defaweux
Vrije Universiteit Brussel
Faculty of Science - 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
Center for Emergent Computing
10, Collington Road - Edinburgh, United Kingdom
j.van.hemert@napier.ac.uk

Johan Parent
Vrije Universiteit Brussel
Faculty of Applied Science - ETRO
Pleinlaan, 2 - Brussels, Belgium
johan@info.vub.ac.be

Abstract- This paper proposes an evolutionary approach for the composition of solutions in an incremental way. The approach is based on the metaphor of transitions in complexity discussed in the context of evolutionary biology. Partially defined solutions interact and evolve into aggregations until a full solution for the problem at hand is found. The impact of the initial population on the outcome and the dynamics of the process is evaluated using the domain of binary constraint satisfaction problems.

1 Introduction

Stochastic Local Search Optimization covers a lot of optimization techniques which aim to traverse the solution space in the most efficient way [Hoos and Stützle, 2005]. These techniques, such as evolutionary algorithms (EA), evolve representations of fully qualified solutions and try to identify which part of the search space is more likely to contain the best one.

As an alternative to evolving completely specified solutions, this paper proposes to grow incrementally partial solutions using a biological model as inspiration: Evolutionary transitions in complexity [Maynard Smith and Szathmáry, 1995, Michod, 1999]. The Transition model we propose is a continuation of previous attempts to introduce a collaborative framework into evolutionary algorithms [De Jong and Potter, 1995, Potter, 1997, Watson and Pollack, 2000, Watson and Pollack, 2002]. Our approach tries to mimic the principle of the biological transitions in generational evolutionary search algorithms where the emergence of complexity is the result of the evolutionary search process rather than the result of a manual "divide and conquer" approach.

In this model, we consider partially defined solutions that specify just an incomplete solution to the problem at hand. As a consequence, these partially defined solutions cannot hope to solve the complete problem unless they collaborate with each other. When beneficial collaborations are identified, a transition can occur, i.e., they are aggregated into a new (more complex) solution. This aggregation can represent in turn a partially or fully defined solution. This transition operates therefore as an aggregation mechanism or, to refer to the biological counterpart, as the fixation of a symbiotic relation.

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 [Tsang, 1993, Achlioptas et al., 2001] and therefore, it can be used as a firm benchmark for comparing with other evolutionary techniques [van Hemert, 2002, Craenen et al., 2003]. 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 benchmark our Transition model. Third, and most importantly, it can easily be described in terms of the aggregation of simple solution units. The idea of aggregating simple solution units was used before to speed up constraint programming techniques by learning which aggregations are undesirable, hence, should be avoided [Frost and Dechter, 1994]. Here we shall take the opposite approach whereby we evolve aggregations which contribute to solving the problem i.e., aggregations that are desirable.

In previous work [Defaweux et al., 2005] the Transition model described here was successfully tested against other evolutionary techniques for BINCSP. The model succeeded in incrementally building solutions for the problems at hand. In this paper, we will test the sensitivity of the Transition model to the initialization of the population of partially defined solutions. Studying the impact of initial population on the outcome of an evolutionary algorithm can eventually help us draw conclusions about the algorithm limitations. If no significant difference between various initial population setup can be observed, then, we can conclude that our algorithm can be applied to the problems without the need of preliminary parameter tuning. In the latter case, we can say that the algorithm is robust toward initial condition setups. We will first introduce shortly BINCSP and the transition model. Then, we provide the reader with the results of empirical research on this model given two different categories of population setups i.e randomly defined and sound populations.

2 Related Work

The first Genetic Algorithms (GA's) that worked with partially defined solutions were referred to as Messy GA's [Goldberg et al., 1989, Goldberg et al., 1990]. The idea behind Messy GA's was to explicitly evolve the building blocks of a solution instead of doing this implicitly in

the completely defined, fixed-size representation. Since Messy GA's worked with a variable length genotype, the crossover operation was adapted into an equivalent operator for variable length representation. Moreover, the issues of over- and under-specification when using a variable length representation were introduced in that context. Over-specification referred to the fact that two or more values are assigned for the same position. Under-specification referred to the possibility that the solution did not contain a value for a particular position in the representations. Messy GA's are related to our approach in the sense that both work with variable length representations and that one has to deal with issues of over- and under-specification and reproduction. Messy GA's do however not address the question of collaboration between partially defined solutions, whether the aggregation of units is the result of a good collaboration or not and how complexity arises in an emergent way.

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 [De Jong and Potter, 1995, Potter, 1997]. The divide and conquer approach however is the result of engineering techniques and lacks the principle of emergence of the complete solutions through the interactions of simple ones.

Other techniques to promote collaboration exist. Among them, an interesting approach is given by the so called Multi-Level Selection models (MLS) [Lenaerts et al., 2002, Lenaerts et al., 2003]. In this model, solutions are spread into groups in which they evolve and these groups can then either disappear at each generation or maintain themselves indefinitely. The groups as a whole can then identify partial or complete solution to the problem. Hence, evolution by MLS provides a mechanism to identify collaborative units. Because of this benefit, we are now designing a system that combines MLS with the Transition model in order to improve the search process.

Finally, the symbiogenetic model [Watson and Pollack, 2000, Watson and Pollack, 2002] 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 parts 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 which solve nicely a part of the problem are more likely to be selected for the next generation. There is however a serious cost to 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. Finally, Watson works with initial sound population and considers this population in the context of an ecosystem. As the evolutionary process goes on, new solutions may be added to the population but no partial solution ever disappear as the consequence of selection. This way, the symbiogenetic model of Watson guarantees that the system may eventually find a solution as the right combination may always occur. However, the requested time for this solution to emerge can be quite expensive as the population to work on becomes larger over time.

3 Binary Constraints Satisfaction Problems

Constraint Satisfaction Problems (CSP) [Tsang, 1993] form a NP-complete problem class where, on the one hand, one has 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 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 [Rossi and Dhar, 1990].

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. Also, for each pair of variables, only one binary constraint may be defined.

The problem involves finding the correct assignment for the variables so that all these constraints are satisfied. 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 assignment 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 Models

In this section, we will illustrate the key features of our transition model through a description of the simple example described in Equation (1).

4.1 Preliminary Definitions

We first introduce the reader with some definitions we will use throughout the model description:

- A *Partial Solution* is the assignment of a subset of the variable set. An example of a partial solution is given by (2).

- A *Solution* is the particular case where assignments are defined for the entire variable set.
- A *Genotype* is the representation of the assignment of variables. 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 is linked. (An example is given by (3)).
- An *(induced) phenotype* of a solution is the variable assignments one obtains when working out the symbiotic relation of the solution with its symbiotic partner. The way such a phenotype is obtained is explained in the next section. If we denote by s the solution and by *sp* its symbiotic partner, we will denote the induced phenotype of s linked with *sp* by $\phi(s, sp)$.

4.2 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 is for example:

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

(2) is called the genotype of the solution. A solution is *fully qualifying* in our example problem when assignments are defined for all six variables.

One of the basic requirements of incremental search is to define how these partially defined solutions interact. This is explained using the following example. Let solution s defined by (2) interact with a symbiotic partner sp defined by $(\langle 3, 1\rangle, \langle 2, 3\rangle)$[1]. We call this interacting partner the symbiotic partner of (2) as a 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 the sharing of information between the partners. The outcome of the information sharing between a solution and its symbiotic partner is called the phenotype of the solution. For example, the way the phenotype of the solution with the symbiotic partner described in (3) is obtained follows these steps:

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

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

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

$$\langle 1, 1\rangle, \langle 3, 2\rangle, \langle 2, 3\rangle \tag{5}$$

Conflicting values are equivalent to over-specified representations in [Goldberg et al., 1989]. Our strategy to solve this over-specification problem corresponds to the probabilistic strategy proposed there.

(5) is called the (induced) phenotype of the partial solution (2). This phenotype is used for evaluation and the result of the evaluation is assigned to the genotype (2). We denote the phenotype of a solution s interacting with a symbiotic partner sp by: $\phi(s, sp)$. The phenotype assigned to the symbiotic partner is obtained the same way. Yet, the policy about conflicting values may yield another representation than the one we obtained for the initial partial solution. This asymmetry between the phenotype of a solution and the phenotype of its symbiotic partner increases the chance for the exploration of the evolutionary process.

4.3 Evaluation

In the context of BINCSP, we will consider two types of evaluation. The first type of evaluation considers the quality of the partially defined solution with respect to the entire constraints set. This evaluation function corresponds to the classical approach of fitness computation for CSP solving EA's. That is the ratio of constraints from the constraints set that are satisfied by the solution. The second type of evaluation only considers 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 constraint 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 do not violate 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 constraints set, $|C|$ the size of the constraints 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 relatively to the entire set of constraints. It does however not give any indication of the quality of the partially defined assignments *relatively to the constraints it covers*. To see whether an association works fine or not, we therefore define a restricted fitness measure that only considers the constraints covered by the phenotype of the solution. If we denote by $cov(s, C)$ the set of constraints

[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

covered by s, the covering fitness measure is given by:

$$f_{cov}(s) = \frac{1}{|cov(\phi(s, sp), C)|} \sum_{k \in cov(\phi(s,sp),C)} c_k(\phi(s, sp)) \quad (8)$$

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*.

Note that f_{cov} is related to the strategy adopted by Messy GA's for dealing with under-specification. However, in our model, this covering fitness is not used to guide the evolutionary process. It is used as an observer that decides when a new level of selection emerges.

4.4 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, with a small probability of mutation. When a solution is selected, it will therefore be replicated into a new solution. At the same time, the selected solution can help its symbiotic partner to replicate. This replication of the symbiotic partner is currently random based, alternatively it could reflect a more problem specific strategy. 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 working units can survive over more than one generation. The idea of performing this replication in group is based on our previous work in the context of multi-level selection [Lenaerts et al., 2002, Lenaerts et al., 2003].

There is a special case where the result of the interaction will be used for replication rather than the genotype of a solution. This special case occurs when the phenotype happens to solve the sub-problem defined by the covering set of constraints. That is, when f_{cov} is greater than a certain threshold value. In this latter case, the transition creates a new solution at a higher complexity level. Here we define complexity as the number of assigned variables in the solution.

Let us take the example solution previously discussed. The solution described by (2) with the phenotype given in (5) has 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 expression 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 key of our incremental approach. Solutions are incrementally grown according to their success in solving the sub-problem they define.

4.5 About transition models

Although our model was adapted to an optimization framework, it still shares important properties with the biological Transition model on which it was based.

First, we do not use the standard genetic algorithm operator for recombination, i.e., crossover. Instead, we use a simple replicator model where the individuals (in our case partial-solutions) are only concerned with copying themselves. This replication process is believed to be at the origin of complexity in biological systems.

Second, we suppose that all partial solutions are collaborating in order to find the solutions. This is inspired by a biological claim that requires cooperation as a prerequisite for the emergence of complexity [Michod, 1999]. As we are interested in building complex solutions, we therefore enforce cooperative units. This is a little different from the biological world where all kinds of interactions are considered and where the question of emergence of cooperation is addressed. We are not concerned (at this time) with how cooperation emerges between the solution units. We suppose that these units are already cooperating as we wish them to build complex entities. As indicated in Section 2, this issue will be investigated when the multi-level selection framework is combined with the Transition model.

Finally, the information sharing strategy we adopted allows information to be exchanged when values are conflicting on certain allele (such as this was the case in example (4) on variable 1). When fully grown genotypes are created, this strategy corresponds to a simple uniform crossover operation as it is the case in GA's. However, this crossover operation is simulated as full grown solutions emerge. When partial solutions interact, the exchange can only happen on conflicting parts of the genotype, yet, preserving the non conflicting parts of the genotypes. We can therefore see that our Transition model is able to mimic uniform crossover but combines this with a preservation mechanism that prevents the destruction of good working parts of the solutions.

5 Validation

The validation of this model has been performed against three evolutionary techniques [van Hemert, 2002]:

- Co-evolutionary Constraints Satisfaction (CCS)
- Micro-genetic algorithm with iterative descent (MID)
- Stepwise Adaptation of Weights (SAW)

The results were described in [Defaweux et al., 2005] and the conclusion was that the model succeeds in evolving fully specified solutions by incrementally aggregating partial units. Furthermore, comparing the Transition model with the three problem-specific evolutionary techniques above reveals that the Transition performs well against these techniques without the need to add problem-specific feature to it. Hence, this opens an interesting door toward the hybridization of the Transition model with problem-specific heuristics.

So far, no test was performed to check how sensitive the Transition model is with respect to its initial parameter setup. In this paper, we discuss the robustness of the model regarding the initial population setup. Other parameters such as mutation rate or eventually adaptation of the

aggregation mechanism will be considered in further studies.

6 Experimentation

6.1 Goal

The way the initial population is generated influences the outcome of an evolutionary algorithm. We compare two different setups for the initial population and see whether it influences the outcome of the evolutionary process as well as its dynamics. We will consider the two following initial condition setups (which lie at the 2 extreme possible setups):

- A uniform random generated population of 100 partial solutions. The initial partial solutions represents the assignments of only two distinct variable. As such, they can expect to solve certain constraints and therefore receive a fitness greater than zero without the need for collaboration. However, restricting the population to 100 partial solutions makes it impossible for all assignments to each variable to be present in the population in the beginning.
- An initial *sound* population of elementary solution units. By elementary solution unit, we understand the assignment of exactly one variable and by sound population we assume that all possible assignments for each variable are present. Since an elementary solution unit defines exactly one variable, it can not have a fitness greater than zero unless it collaborates with at least one other elementary unit. This follows from the constraints, which are defined on exactly two variables. For a BINCSP of 15 variables each of domain 15, this result in an initial population of 225 solution units.

6.2 Experimental setup

We use randomly generated problem instances of BINCSP. The RandomCSP package [van Hemert, 2004] is used to generate the suite of test problem instances [van Hemert, 2002]. To scale the difficulty of the problem instances, these CSP are generated according to two parameters. The first parameter is the density of the BINCSP:

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

This parameter reflects how many constraints there are relatively to the maximum amount of constraints there can be.

The second parameter is the average tightness $\bar{p}_2$ that reflects the average complexity of the constraints. That is, how many invalid simultaneous assignments of two variables are allowed in one constraint, averaged over all constraints. If we denote by $|c|$ the number of assignments in the constraints c that satisfies c, and by m the domain size of the variables in constraints c, we have:

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

Density p_1	Tightness 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 Evaluations for random generated population

p_1 and $\bar{p}_2$ are called "order parameters", as they can be used to order the problem instances of a class of problems. By keeping the number of variables and the domain size of each variable constant, and then varying these parameters we can induce a phase transition [Cheeseman et al., 1991]. 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 the location of the hardest to solve (or to proof unsolvable) problems. This gives rise to the typical easy-hard-easy pattern.

6.2.1 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 generated per combination of p_1,$\bar{p}_2$, we proceed with 10 runs, each run using a different seed, This yield a total of 250 runs for each parameter setup. The maximum amount of evaluations is set to $100\,000$. We are interested in generating solvable instances so that we can compare the success ratio in both setups. As a consequence, we did not generate problem instances for high values of p_1 and $\bar{p}_2$ since solvable instances either do not exist or cannot be found in a reasonable amount of time.

We have tested this setup with the two types of initial populations described earlier.

6.2.2 Observations

We are interested in observing following information:

- Solution sizes: We observe the maximum and minimum size of the solutions over time, together with the size of the intersection of the solutions with their symbiotic partner, i.e. the number of assignments for the same variable present in both interacting solutions. From this intersection, we distinguish conflicting values from others, that is, values for the same variable which differ in both partial solutions.

Density p_1	Tightness 0.1	0.3	0.5	0.7	0.9
0.1	1.0 (4)	1.0 (5)	1.0 (7)	1.0 (13)	0.99 (3627)
0.3	1.0 (5)	1.0 (12)	1.0 (43)	0.28 (11758)	- -
0.5	1.0 (6)	1.0 (25)	0.54 (11409)	- -	- -
0.7	1.0 (8)	1.0 (61)	0.29 (9763)	- -	- -
0.9	1.0 (11)	0.97 (2335)	- -	- -	- -

Table 2: success ratio and Average Number of Evaluations for sound population

	Comparison of success-ratio	Wilcoxon Mann-Whitney rank-test
$p_1 = 0.3$,$p_2 = 0.9$	0.3603	< 0.0001
$p_1 = 0.5$,$p_2 = 0.5$	0.4735	0.2065
$p_1 = 0.5$,$p_2 = 0.7$	0.4953	0.5887
$p_1 = 0.7$,$p_2 = 0.3$	0.0609	0.07869
$p_1 = 0.9$,$p_2 = 0.1$	0.0926	< 0.0001

Table 3: Significance in the difference between results for initial setups for difficult problems: the first column gives the p-values for the test of equivalence between the success ratio of both initial setups, the second column gives the p-value for the Wilcoxon-Mann-Whitney rank test between both setups for the equivalence of the distribution of the number of required evaluations to reach a solution.

- Fitnesses: 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 of Evaluations to reach a solution: this gives a measure of how fast our algorithm can reach the goal (the solution).

This information can be split up in two categories. The first one helps us to observe the dynamics of the evolutionary process (evolution of solutions sizes and fitnesses) and the second category helps us to compare how well the two initial populations perform (success ratio and average amount of evaluation needed to reach the solution).

6.3 Results

6.3.1 Comparing Success Ratio and Average Evaluation Steps

In Table 1 and 2, we can observe the success ratio and the average number of evaluation steps needed to reach a solution for random generated population and sound population, respectively . In Table 2, we have underlined the success ratio of the sounded population whenever it performs better than the uniform randomly generated population. We can see that, in the overall, the initial population has little effect on the chance of success, which is interesting as it indicates that our algorithm performs well with a small initial population and hence does not need a big and complete population to achieve good results.

For very easy problems, that is, problems where solutions are found on average in less than 10 steps, the randomly generated population finds it a little faster. This is probably due to the fact that the initial partial solutions are already aggregated in pairs, speeding up the convergence toward the solutions. However, when the problem requires a little more exploration, yet remaining easy, sound populations are faster as they can combine good units from scratch and need not to solve conflicts as the initial pairs in the random generated population are requested to do. However, for very difficult problems ($p_1 = 0.5$, $p_2 = 0.5$; $p_1 = 0.7$, $p_2 = 0.5$; $p_1 = 0.3$, $p_2 = 0.7$; $p_1 = 0.1$, $p_2 = 0.9$) where we observe that the sound population performs as well or slightly better than the random generated population, we can see that the number of evaluations needed is higher for the sound population. An explanation for this may lie in the initial exploration process. The random generated population already starts with pairs of solutions; if good pairs are present, they will be favored over the others by the fitness proportionate selection. The sound population however needs to build these basic pairs from scratch. It will therefore favor a greater exploration in the very beginning of the process. This yields slightly better results when considering the success ratio but this exploration has a price as it converges more slowly to a solution.

6.3.2 Significance of the initial population setup

Table 1 and Table 2 show little difference between the two initial setups. To assess whether these differences are significant or not, we applied a two samples test to the result set obtained with each population setup for the difficult problem instances. The tests will eventually not allow us to reject the hypothesis that both process show an equivalent behavior. Since we could not suppose that our sample follows a normal distribution, we chose for the Wilcoxon-Mann-Whitney test [Barlow, 1995].

The first column of Table 3 gives the p-value of a two samples test for the comparison of the success ratio obtained with each initial population setup. The second column provides us with the p-value of the Wilcoxon-Mann-Whitney two sample rank test that checks whether the distribution of the number of evaluations to reach a solution between both initial population setups are equivalent. We can see that for a significance level of 5%, the equivalence in the success ratio can not be rejected for any of the test case. In other words, we cannot conclude that the initial population setup yields significantly different chance of success in both cases. Furthermore, the necessary computational time, measured in number of evaluations, to obtain a solution is not significantly different either, when considering the very difficult test cases (test-cases with a success ratio < 0.7).

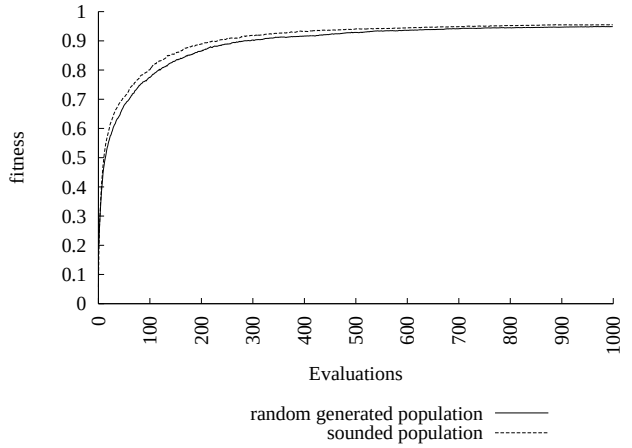

Figure 1: Evolution of the fitnesses for random generated population and sound population

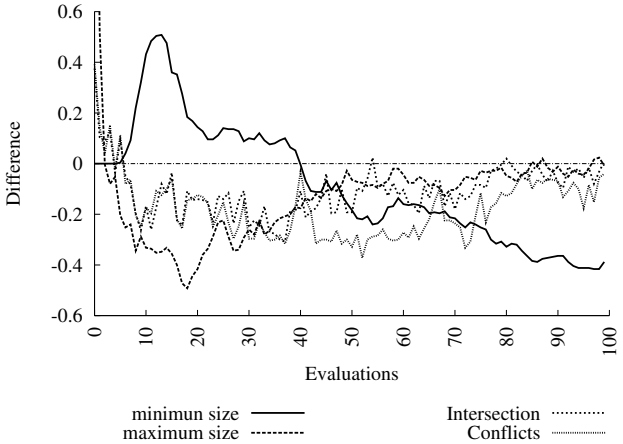

Figure 2: The difference in evolution of the genotype sizes

However, there exists a significant difference when considering not too difficult instances (such as $p_1 = 0.9, p_2 = 0.3$ and $p_1 = 0.1, p_2 = 0.9$). From these statistics, we may suppose that the chance of success are not significantly different from one population setup to another and that the speed of the algorithm is probably not affected by this initial setup when considering very difficult instances. To study these similarities in more details, we will now examine the dynamics of the search process on one of the difficult test cases.

6.3.3 Comparing Dynamics

To compare the dynamics for both initial population setups, we chose to illustrate it on an extremely difficult BINCSP ($p_1 = 0.7$ and $p_2 = 0.5$). This difficult class of problems was chosen because it is where the greatest exploration is required and therefore with high probability will show greater differences between the two population setups. In Figure 1, we can observe the evolution of the best fitness for both population setups. In Figure 2 we show the difference between the random generated population and the sound population for the averages of the maximum size, minimum size, the size of the set of variables shared by both the partial solution and its symbiotic partner (we refer to this set as the intersection), and the size of the set of conflicting values in this intersection. Figure 3 shows the evolution of these sizes for the sound population initial setup.

The fitness evolution confirms the fact that even if sound populations perform slightly better in general, the dynamics of both process are very similar and both initial population setups succeed in solving over 90% of the constraints on average. Figure 2 confirms the similarity in the evolutionary dynamics and illustrates that this similarity is present from the very beginning of the evolutionary process. We can observe that the sizes follow the same trends as the difference remains low (as it never exceeds 1.0 in absolute value). The difference between both population setups is therefore not significantly high and this confirms our hypothesis of robustness of this model toward initial population setup.

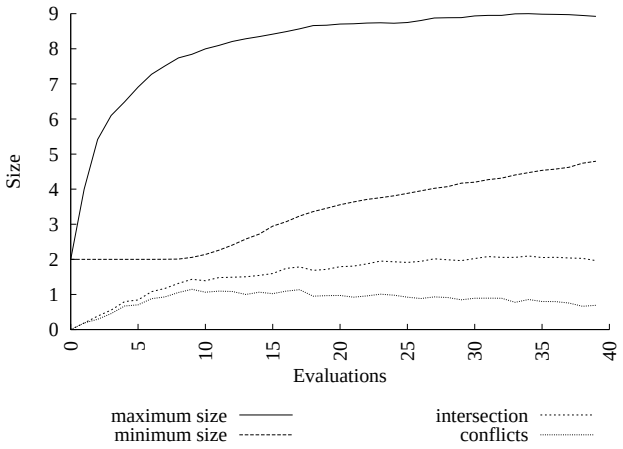

Figure 3: Evolution of the genotype sizes for the sound population

In Figure 3, we plot the evolution of the sizes for 40 generations. We observe that the maximum size converges quickly towards a value which is smaller than the size of a fully defined solution (solution are fully defined when the 15 variables of the problem instance have been assigned). This convergence toward a partially defined solution illustrates the difficulty for the process to find a solution and the necessity for the solutions to cooperate with others to find a good compromising solution. What can be said considering the evolution of the fitness on 1000 steps in Figure 1 and the evolution of sizes in 40 steps in Figure 3 is that if the fitness keeps increasing over time, the maximum size seems to converge relatively fast toward a maximum value. The size graph only covers the 40 first steps and therefore cannot be compared with the fitness graph. However, we can conclude from this that the increase in fitness we observe in Figure 1 is the result of (re)combination through the symbiotic relation. Furthermore, on average, the sound population requests more time to achieve a solution, which illustrates the fact that the sound population succeeds in maintaining sufficient diversity among the population to allow exploration to occur.

7 Conclusion

In this paper, we propose an incremental evolutionary algorithm based on the Transition metaphor described in Biology. We apply this model to various instances of the Binary Constraints Satisfaction Problem with different properties to test whether this incremental model is sensitive to its initial population conditions. We use solvable BINCSP instances scaling from very easy to very difficult.

We consider two methods for creating the initial populations. First, a classical initial EA population consisting of 100 partially defined solutions and second, a sound population consisting of all possible variable assignments as elementary solution units.

From our empirical results we deduce that, on the overall, a sound population achieves slightly better results than a randomly generated population, but this difference is not significantly different. Also, when observing the dynamics of these processes on a particularly difficult problem, we observe that the dynamics are similar. Therefore, we conclude from this that the overall improvement of the sound population over the random generated population is not sufficiently high to stress that the initial population has an impact on the evolutionary process. This result is quite interesting as it shows that our transition model is robust against the initial population setup.

Bibliography

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

[Barlow, 1995] Barlow, R. (1995). *Statistics: a guide to the use of statistical methods in the physical sciences*. John Wiley & Sons.

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

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

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

[Defaweux et al., 2005] Defaweux, A., Lenaerts, T., van Hemert, J., and Parent, J. (2005). Transition models as an incremental approach for problem solving in evolutionary algorithms. In *the Genetic and Evolutionary Computation Conference*, page not known yet. to be published.

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

[Goldberg et al., 1989] Goldberg, D., Korb, B., and Deb, K. (1989). Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, 3:493–530.

[Goldberg et al., 1990] Goldberg, D., Korb, B., and Deb, K. (1990). Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4:415–444.

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

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

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

[Maynard Smith and Szathmáry, 1995] Maynard Smith, J. and Szathmáry, E. (1995). *The major transitions in evolution*. Oxford University Press.

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

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

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

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

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

[van Hemert, 2004] van Hemert, J. (2004). RandomCSP. Freely available from `http://freshmeat.net/projects/randomcsp/`.

[Watson and Pollack, 2000] Watson, R. A. and Pollack, J. B. (2000). 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.

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