

Template Design using Extremal Optimization with Multiple Search Operators

Raymond Chiong*, Thomas Weise[†], and Bee Theng Lau*

*School of Computing & Design, Swinburne University of Technology (Sarawak Campus), Kuching, Malaysia

[†]Nature Inspired Computation and Application Laboratory, University of Science and Technology of China, Hefei, China

Abstract—The template design problem is a constrained optimization problem originated from the printing industry. It involves printing several variations of a design onto one or more stencil sheets, where the aims are to minimize the number of stencils as well as the overproduction of prints of a particular design. Over the years, exact solution methods have been used to solve the problem. These methods could be useful for small to moderate-sized problem instances. However, when the problem instances are huge, the search space may easily grow too large for the systematic approaches. To date, no meta-heuristic or soft computing techniques have been used for this problem. In this paper, we propose the use of Extremal Optimization (EO) with multiple search operators for solving the template design problem. Different combinations of the search operators are tested via extensive numerical experiments. The results show that EO is indeed a feasible approach for template design optimization. The hybridization of EO with a deterministic local search has proven to be particularly effective.

I. INTRODUCTION

Template design is a practical problem which was brought to the attention of the Operations Research community by a small printing company. Formalized by Proll and Smith [1], it has since been solved with integer linear programming (ILP) and constraint satisfaction programming (CSP) approaches. Exact solutions have been produced for the instances presented in the original work. Deterministic algorithms such as ILP and CSP search through the search space in a structured way, which is time consuming for large search spaces. Besides that, when more complicated instances are used, the search space may easily grow too large for these systematic approaches.

Extremal Optimization (EO) has first been devised by Boettcher in 1999 [2] to harness extremal dynamics to facilitate the search for spin glass ground states. A special case of EO has been conceived separately as a means of solving hard satisfiability problems by Selman et al. [3]. Essentially a stochastic local search, EO allows for hill climbing with a probability that declines exponentially with the fitness rank of the proposed new solution, rather than using the raw fitness. As such, it is a good choice for solving assignment problems, which it has been applied to before [4].

Another group of authors has successfully applied it to multi-objective optimization in [5, 6]. In response to these encouraging results, we have adapted the algorithm to template design, introducing some interesting enhancements.

II. THE TEMPLATE DESIGN PROBLEM

Template design optimization has its application in the printing industry, where product flyers or product containers have to be printed on paper or thin cardboard. The printing press uses “stencil” sheets which accommodate the templates of the items to be printed. The flyers or containers may be needed in different colors and with different descriptions, according to slight product variations, but are usually of identical size. A practical example is a commercial order for cat food cartons, where the cat food comes in several flavors.

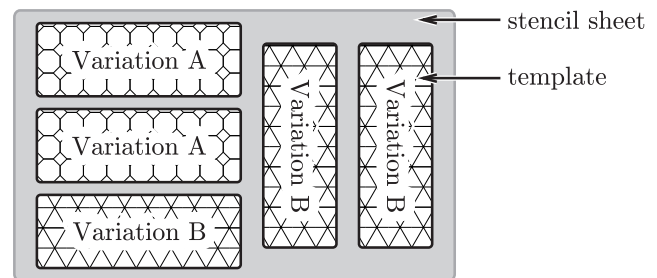


Fig. 1. Templates for different variations of prints laid out on a stencil sheet. Given the size of the templates, this stencil has five slots.

Orders for prints may contain different quantities for each variation of the print, as shown in Fig. 1. Each variation has to be represented by at least one template on one or more of the stencil sheets. As stencil sheets are expensive to produce, one of the optimization criteria is minimizing the number of sheets used. To satisfy a customer, print variations have to be created in at least the ordered quantities. Depending on the layout of the stencils and the types of templates included, it is likely that the minimal number of pressings per stencil, given by the ordered quantities, will cause overproduction of one or more of the print variations. Minimizing this overproduction is the second criterion, for which it becomes necessary to optimize the number of prints per stencil.

To date, few attempts at solving this problem can be found in the literature. Apart from the original ILP and CSP by Proll and Smith [1], Nielsen et al. [7] used the template design problem for testing the transformation of constraint satisfaction formulations to ILP models. Prestwich et al. [8] developed a variable-weighting integer linear programming (VWILP) model for a variation of Nielsen et al.’s problem

Table I
EXAMPLES FOR THE TEMPLATE DESIGN PROBLEM FROM [1].

Problem	Slots per Var. Template	Order Quantities in Thousands	
Cat food	9	7	250, 255, 260, 500, 500, 800, 1100
Herb cartons	42	30	60, 60, 70, 70, 70, 70, 70, 70, 80, 80, 80, 80, 90, 90, 90, 90, 90, 100, 100, 100, 100, 150, 230, 230, 230, 230, 280, 280
Magazine inserts	40	50	50, 53, 55, 60, 85, 90, 100, 100, 105, 110, 137, 140, 140, 140, 150, 150, 150, 150, 150, 150, 150, 150, 168, 170, 170, 195, 195, 200, 200, 200, 210, 210, 225, 230, 230, 230, 250, 250, 250, 250, 250, 250, 250, 265, 270, 270, 375, 375, 405

formulation. Bender’s Decomposition approach, a method based on optimizing variables separately, was later extended by Tarim and Miguel [9] to be applicable to stochastic constraint programs with linear recourse. More recently, Prestwich and Verachi [10] compared the VWILP and Bender’s approaches with their bounds-oriented local search (BOLOS), a hybridization of a constraint programming approach and a constructive local search. To the best of our knowledge, none of the known approaches in the literature applies meta-heuristics or soft computing techniques to the template design problem. In this paper, the problem instances used are taken from [1] and reproduced in Table I.

III. SOLUTION STRUCTURE AND SEARCH OPERATORS

Two of the most important aspects of our EO approach are the solution structure and the search operators, which will be discussed next.

A. Solution Structure

A candidate solution x for the template design problem with v different variations and templates with S number of slots consists of two structures: the list of template definitions $x.T$ and the list $x.R$ containing how often the templates are pressed. The template $x.T_i$ is pressed exactly $x.R_i$ times. We will refer to the index i as *locus* since we consider the tuples $(x.T_i, x.R_i)$ to be the genes of the candidate solutions. $x.T_i$ is, in turn, a tuple of size v and the element at index j , $x.T_{i,j}$ denotes how often a blueprint of variation j is present on template i . The search space \mathbb{X} comprises all valid candidate solutions, i. e., those for which the following equations hold:

$$x \in \mathbb{X} \Rightarrow x.R_i > 0 \quad \forall i \in 1..\text{len}(x.R) \quad (1)$$

$$x \in \mathbb{X} \Rightarrow \left[\sum_{j=1}^{\text{len}(x.T_i)} x.T_{i,j} \right] = S \quad \forall i \in 1..\text{len}(x.T) \quad (2)$$

B. Search Operators

All the search operators we use ensure candidate solutions are always valid. A search operation “op” has the form $\text{op}(x, L, ar) \mapsto x' \in \mathbb{X}$ where x is the candidate solution to be processed, L is the *locus* selected for modification according to the gene fitness, ar is the archive of non-dominated candidate solutions already discovered, and x' is the new candidate solution created by the operator. The search operators can be divided into two classes: those with minor impact and those which heavily modify a solution structure.

1) Small-Impact Operators:

Modify the Number of Pressings: Modify the number of pressings of a template at index L by setting $x'.R_L$ to a random number normally distributed around $x.R_L$.

Modify the Template: Modify the template at index L by shuffling quantities between the variations.

2) Large-Impact Operators:

Create the Initial Configuration: The initial solution is created by adding new template i in a loop until all orders are satisfied.

Add Template: Add a new, randomly configured template to the candidate solution; applicable if $\text{len}(x.T) < 10$.

Delete Template: The template at index L is removed from the candidate solution and its pressings are randomly distributed amongst the remaining templates; applicable if $\text{len}(x.T) > 1$.

Split Template: Select one of the templates in x and split it into two similar templates amongst which the original pressings number is divided; applicable if $n < 10$.

Crossover: Combine the current candidate solution x with a different one randomly drawn from the archive of non-dominated solutions. From both candidate solutions, random templates are selected; applicable if $n < 10$ and $\text{len}(ar) > 1$.

Draw Non-Dominated Candidate Solution: The current candidate solution is discarded and replaced with a solution (uniformly) drawn from the archive of non-dominated solutions; applicable if ar is not empty.

IV. FITNESS MEASURES

There are two types of utility measures: the global (solution-centric) objective functions f and the local (gene-centric) fitness measures.

A. Global Fitness Measures

The template design problem is inherently multi-objective, meaning that we have to deal with a vector function $\vec{f} = (f_1, f_2, f_3) : \mathbb{X} \mapsto \mathbb{R}^3$ instead of a single criterion $f : \mathbb{X} \mapsto \mathbb{R}$. In multi-objective problems, usually the notation of Pareto domination is applied: A candidate solution $x_1 \in \mathbb{X}$ dominates another one ($x \in \mathbb{X}$) if and only if it is better in at least one optimization criterion and not worse in any other. The domination relation denoted by $\vec{f}(x_1) \preceq \vec{f}(x_2)$ defines a partial order on the search space. The goal of

optimization is to find all candidate solutions x^* which are not dominated by any other point in the search space, i. e., $\neg \exists x \in \mathbb{X} : \vec{f}(x) \preceq \vec{f}(x^*)$.

1) *Number of Templates* f_1 : In order to fulfill all orders, a single template will usually not suffice. We therefore introduced several search operators able to modify the number of templates in a candidate solution in Section III-B2. Producing a template, however, is costly and thus, the number of templates used should be minimized. As such, we define the first objective function as:

$$f_1(x) = \text{len}(x.T) \quad (3)$$

2) *Maximum Deviation* f_2 : The template design problem is centered around fulfilling a customer order for pressing v variations of a certain design. It will rarely be possible to produce exactly the ordered quantities Q_j . Thus, for each design j , the number of actually produced units may deviate (relatively) by Δ_j from Q_j :

$$\Delta_j(x) = \frac{\left[\sum_{i=1}^{\text{len}(x.T)} x.T_{i,j} * x.R_i \right] - Q_j}{Q_j} \quad (4)$$

Negative values of $\Delta_j(x)$ denote underproduction, positive mean an overproduction of the specified variation. The second objective function f_2 , again subject to minimization, corresponds to the worst (absolute value of the) relative deviation:

$$f_2(x) = \max_{v,j \in 1..v} |\Delta_j(x)| \quad (5)$$

3) *Mean Deviation* f_3 : f_2 provides information about the ‘‘biggest problem’’ in a candidate solution but does not represent its overall quality. A good measure for this is the mean relative deviation f_3 :

$$f_3(x) = \frac{1}{v} \sum_{j=1}^v |\Delta_j(x)| \quad (6)$$

As can be seen, the three objectives are at least partly contradicting. A high precision in order fulfillment (i. e., good f_2 and f_3 values) can be attained by utilizing more templates which leads to a degeneration in f_1 . A low maximum deviation f_2 does not necessarily lead to a low mean deviation f_3 and vice versa.

B. Gene Fitness Measures

EO focuses on modifying parts of a candidate solution according to their estimated contribution to the overall fitness. Therefore, gene-centered fitness measures are needed besides the solution-centered objective values. We define the fitness measures g as follows:

1) *Number of Pressings*: The contribution of the template at index i to the overall fitness of a candidate solution depends on how often it is pressed. Templates for which $g_1(x, i) = x.R_i$ is low thus have a low overall value.

2) *Contribution to Deviation*: With g_2 we try to approximate the contribution of a template to the overall deviation f_3 . We therefore define the proportion $pc(x, i, j)$ to which template i adds to the overall production of a variation j in candidate solution x in Equation 7. The deviation contribution $\delta(x, i, j)$ to the production of the variation j follows in Equation 8. If we have an underproduction of variation j , $\Delta_j(x)$ is negative and $0 \leq pc(x, i, j) < 1$. Since the bigger $pc(x, i, j)$ the better, we add $1 - pc(x, i, j)$, i. e., small production contributions lead to higher (worse) gene fitness values. Vice versa, if $\Delta_j(x) > 0$, i. e., overproduction of variation j , smaller values of $pc(x, i, j)$ should be favored. The $\delta(x, i, j)$ are summed up in the gene fitness in Equation 9. Different from g_1 , the higher its $g_2(x, i)$ is, the worse the gene at locus i is.

$$pc(x, i, j) = \frac{x.R_i * x.T_{i,j}}{Q_j} \quad (7)$$

$$\delta(x, i, j) = \begin{cases} -\Delta_j(x) * (1 - pc(x, i, j)) & \text{if } \Delta_j(x) < 0 \\ \Delta_j(x) * pc(x, i, j) & \text{otherwise} \end{cases} \quad (8)$$

$$g_2(x, i) = \sum_{j=1}^v \delta(x, i, j) \quad (9)$$

V. OVERVIEW OF THE ALGORITHM

An overview of our EO-based algorithm with all the elements specified in Section III and Section IV is shown in Fig. 2.

VI. EXPERIMENTS

As aforementioned, numerical experiments are performed on all the three problem instances listed in Table I.

A. Settings

In our experiments, the impact of five different parameters are tested. The small-impact operators can be used either as a pure EO ($ls = 0$) or as steps in a local search subalgorithm ($ls = 1$). We may either use the template adding operator ($at = 1$) or not ($at = 0$), and the same goes for the template splitting operator ($st \in \{0, 1\}$), the crossover operator ($cr \in \{0, 1\}$), and the operator which randomly draws previously discovered, non-dominated candidate solutions ($ur \in \{0, 1\}$).

We have decided to run full factorial experiments [11] in order to test the utility of these different features. This means that the five parameters avail to a combination of 32 configurations. For each configuration, we performed multiple experimental runs. The maximum number of search steps of each run was limited to at most 200 000 000.

B. Evaluation

As established in Section IV-A, the template design problem is analyzed from the perspective of three objectives, and an archive of non-dominated candidate solutions is kept.

Fig. 2. $ar \leftarrow \text{eoOptimize}(v, S, Q, ls)$

```

1: Input:  $v$ : the number of variations
2: Input:  $S$ : the slots per template
3: Input:  $Q$ : the order quantities
4: Input:  $ls$ : true if local search with small-impact ops
5: Output:  $ar$ : non-dominated candidate solutions
6: begin
7:  $ar \leftarrow \emptyset$ 
8:  $z \leftarrow \text{randUniform}(1..2)$ 
9:  $x \leftarrow \text{create}(\cdot, 0, \emptyset)$ 
10: while  $\neg \text{terminationCriterion}$  do
11:   if  $\text{randUniform}(1..16) = 1$  then
12:      $z \leftarrow 3 - z$ 
13:   end if
14:    $x \leftarrow \text{sort}(x, g_z)$ 
15:    $L \leftarrow \text{randPower}(k^{-1.5} \in 1..\text{len}(x.T))$ 
16:   if  $\text{randUniform}(1..64) > 0$  then
17:      $op \leftarrow \text{random applicable small-impact op.}$ 
18:      $x' \leftarrow op(x, L, ar)$ 
19:     if  $(\vec{f}(x') \preceq \vec{f}(x)) \vee (ls = 0)$  then
20:        $x \leftarrow x'$ 
21:     end if
22:   else
23:      $op \leftarrow \text{random applicable large-impact op.}$ 
24:      $x' \leftarrow op(x, L, ar)$ 
25:      $x \leftarrow x'$ 
26:   end if
27:    $ar \leftarrow ar \setminus \{d : (d \in ar) \wedge (\vec{f}(x') \preceq \vec{f}(d))\}$ 
28:   if  $\neg \exists b \in ar : (\vec{f}(b) \preceq \vec{f}(x'))$  then
29:      $ar \leftarrow ar \cup \{x'\}$ 
30:     //archive pruning if needed
31:   end if
32: end while
33: return  $ar$ 
34: end

```

Usually, factorial experiments would be evaluated with statistical tests two-fold: First, we would determine the best configuration and then establish trends on which parameter has a good influence in general. In the case of the three-dimensional Pareto frontier, this approach is not feasible.

Therefore, we computed scores $SC(C_i)$ for each single configuration C_i reflecting the relative quality of their results in comparison with all other tested configurations. Two configurations C_1 and C_2 can be compared with each other by comparing the Pareto sets ar_1 and ar_2 obtained with them. Instead of using only the results of a single run, we joined the Pareto sets of all runs of a configuration for this purpose. From two such joint sets, we compared every two candidate solutions with each other according to Equation 10. Based on Equation 11, the score of the configuration is increased by one for every candidate solution from the archive of the opposing configuration which it can dominate and decreased by one in the opposite case. Since the frontiers may contain different numbers of candidate solutions, the

score is normalized to $[-1, 1]$.¹ The total score $SC(C_i)$ of a configuration C_i is then the sum of the ‘‘cmpF’’-based comparison with all other configurations.

$$\text{cmpCS}(x_1, x_2) = \begin{cases} 1 & \text{if } \vec{f}(x_1) \preceq \vec{f}(x_2) \\ -1 & \text{if } \vec{f}(x_2) \preceq \vec{f}(x_1) \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$\text{cmpF}(ar_1, ar_2) = \frac{\sum_{i=1}^{\text{len}(ar_1)} \sum_{j=1}^{\text{len}(ar_2)} \text{cmpCS}(ar_1[i], ar_2[j])}{\text{len}(ar_1) \text{len}(ar_2)} \quad (11)$$

High values of SC indicate a configuration which finds better solutions than others. We can now utilize the score SC in two ways: First, we can determine the configuration with the best overall performance. Second, we can determine which of the features of our approach have good or bad influences by simply summing up the scores of all configurations which have the feature(s) turned on.

C. The Cat Food Experiment

1) *Results:* The cat food problem is the simplest bench-

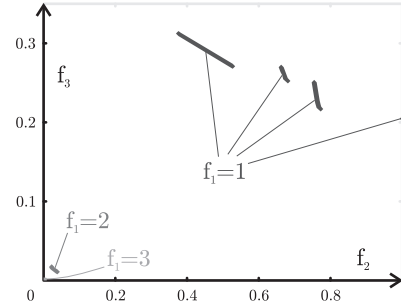


Fig. 3. Joint Pareto frontiers of the cat food experiment: Projection and contours in f_1 .

mark instance of the template design problem. In Fig. 3, we sketched the joint Pareto frontiers of all the experimental runs with all configurations. The frontier for a single template ($f_1 = 1$) is long and discontinuous. For two templates, much better solutions can be found, but there is still a trade-off between maximum and average deviation from the ordered amounts. If three templates can be used, the front becomes very dense and close to optimal.

2) *Trends:* In Table II, we show the best five and worst five EO-configurations for the cat food problem. According to the score values, the settings with the strongest positive influence are using local search ($SC(ls) = 40$) followed by resetting to previously discovered non-dominated candidate

¹Some of the joint frontiers contained more than 100000 candidate solutions. In these cases, a full comparison was not feasible and we performed 1 000 000 random comparisons in order to approximate the exact score instead.

Table II
THE SCORES SC OF THE TESTED CONFIGURATIONS IN THE CAT FOOD PROBLEM.

C	ls	at	st	cr	ur	$SC(C)$
C_{C1}	0	0	1	1	0	3.740
C_{C2}	0	0	0	1	1	3.455
C_{C3}	1	0	0	1	1	3.302
C_{C4}	1	1	0	1	0	3.142
C_{C5}	1	1	1	0	0	3.039
...
C_{C28}	0	0	0	1	0	0.817
C_{C29}	0	1	1	1	0	-3.016
C_{C30}	0	0	1	0	0	-20.753
C_{C31}	0	1	0	0	0	-20.802
C_{C32}	0	1	1	0	0	-21.791

solutions ($SC(ur) = 36$) and using crossover ($SC(cr) = 35$). Interestingly, the template splitting operator seems to have a negative influence ($SC(st) = -15$), even worse than adding randomly configured templates ($SC(at) = -14$).

D. The Herb Cartons Experiment

1) *Results:* The herb cartons problem is a middle-scale

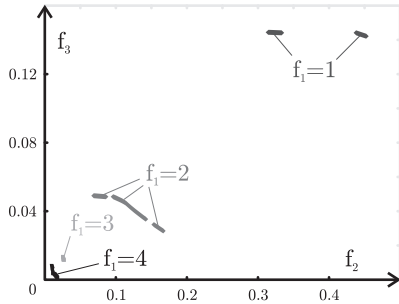


Fig. 4. Joint Pareto frontiers of the herb cartons experiment: Projection and contours in f_1 .

variant of the template design problem. Again, we illustrate the joint Pareto frontiers of all runs (Fig. 4). This time, they are stretched farther and there is also a greater variety between designs consisting of different numbers of templates.

2) *Trends:* Table III lists the best and worst configurations for this instance. Interestingly, none of the best five configurations from the cat food problem made it in the top five of the herb cartons problem. Yet, the general trends are very similar: Again, the settings with the strongest positive influence are using local search ($SC(ls) = 40$) followed by resetting to previously discovered non-dominated candidate solutions ($SC(ur) = 36$) and crossover ($SC(cr) = 35$). Adding randomly configured templates ($SC(at) = -14$) and the template splitting operator ($SC(st) = -15$) again seem to be not really useful.

Table III
THE SCORES SC OF THE TESTED CONFIGURATIONS IN THE HERB PROBLEM.

C	ls	at	st	cr	ur	$SC(C)$
C_{H1}	1	0	1	1	0	9.473
C_{H2}	1	0	1	0	0	9.360
C_{H3}	0	1	0	0	1	9.147
C_{H4}	1	0	1	0	1	8.642
C_{H5}	0	0	0	0	1	8.393
...
C_{H28}	0	0	0	0	0	-9.191
C_{H29}	0	1	1	1	0	-21.455
C_{H30}	0	1	0	0	0	-23.989
C_{H31}	0	0	1	0	0	-24.507
C_{H32}	0	1	1	0	0	-26.081

E. The Magazine Inserts Experiment

1) *Results:* The magazine inserts problem is the hardest among the three. The joint Pareto frontiers of all runs of all configurations are illustrated in Fig. 5. It is interesting to see that there is only little trade-off between the objectives f_2 and f_3 . Fig. 5 shows that the Pareto fronts do not grow with rising f_1 but instead, contract to very small lines.

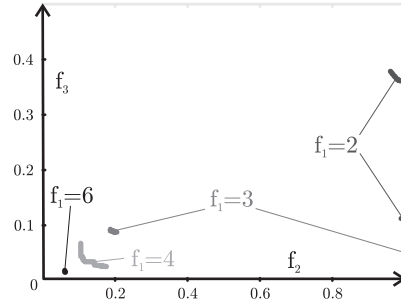


Fig. 5. Joint Pareto frontiers of the magazine inserts experiment: Projection and contours in f_1 .

2) *Trends:* There are significant differences between the performances of the tested configurations. According to the score values, the settings with the strongest positive influence are using local search ($SC(ls) = 84$) followed by crossover ($SC(cr) = 57$) and resetting to previously discovered non-dominated candidate solutions ($SC(ur) = 39$). Interestingly, the configurations using the template splitting operator seems to have a negative influence ($SC(st) = -30$), only topped by the configurations utilizing the operator for adding randomly configured templates ($SC(at) = -45$). In Table IV, the best five and worst five configurations are listed according to their scores.

Surprisingly, the configuration which previously seemed to have performed very well (i.e. the use of additional local search) did not achieve the best result: It has every template-generating operator but the crossover turned on.

Table IV
THE SCORES SC OF THE TESTED CONFIGURATIONS IN THE MAGAZINE
INSERTS PROBLEM.

C	ls	at	st	cr	ur	$SC(C)$
C_{M1}	0	0	0	1	0	13
C_{M2}	1	0	1	0	1	7.920
C_{M3}	1	0	1	1	1	7.832
C_{M4}	1	0	0	0	0	7.815
C_{M5}	1	0	0	0	1	7.585
...
C_{M28}	0	1	0	0	1	-17.777
C_{M29}	0	1	1	0	1	-17.896
C_{M30}	0	1	0	0	0	-19.303
C_{M31}	0	0	1	0	0	-20.031
C_{M32}	0	1	1	0	0	-22.225

The template deleting operator, however, is still present. This leads to a quick convergence to a highly efficient design which consists only of a single template and since – in our implementation – the crossover operator does not mate a candidate solution with itself, no new templates are included. Because of the single, highly efficient candidate solution, the configuration rarely loses any comparison and thus, has a deceptively high score.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have implemented a novel EO algorithm with multiple search operators for the template design problem. We approached the problem with the goal of producing the ordered amount for each design variant as precisely as possible, with as little under- and overproduction as possible. We defined new objective functions, gene fitness functions, and search operators. In a comprehensive, full factorial experimental study, we applied all of them to the three benchmark instances from [1]. We found that some of the operators (local search, resetting to non-dominated individuals, crossover) are especially suitable for this problem while others (adding random templates and template splitting) have little or no benefit at all. Future work will involve the application of a multi-individual EO with deterministic local search within a co-evolutionary framework.

REFERENCES

- [1] L. Proll and B. Smith, “Integer Linear Programming and Constraint Programming Approaches to a Template Design Problem,” *INFORMS Journal on Computing*, vol. 10, no. 3, pp. 225–275, 1998.
- [2] S. Boettcher and A. G. Percus, “Extremal Optimization: Methods Derived from Co-Evolution,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. San Francisco: Morgan Kaufmann, 1999, pp. 825–832.
- [3] B. Selman, H. Levesque, and D. Mitchell, “A New Method for Solving Hard Satisfiability Problems,” in

Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI), San Jose, 1992, pp. 440–446.

- [4] M. Randall, “Enhancements to Extremal Optimisation for Generalised Assignment,” in *Proceedings of the Third Australian Conference on Progress in Artificial Life (ACAL)*, ser. LNCS, M. Randall, H. A. Abbass, and J. Wiles, Eds., vol. 4828. Springer, 2007, pp. 369–380.
- [5] M.-R. Chen, Y.-Z. Lu, and G.-K. Yang, “Multiobjective Optimization Using Population-Based Extremal Optimization,” *Neural Computing & Applications*, vol. 17, no. 2, pp. 101–109, 2008.
- [6] —, “Multiobjective Extremal Optimization with Applications to Engineering Design,” *Journal of Zhejiang University – Science A*, vol. 8, no. 12, pp. 1905–1911, 2007.
- [7] S. R. Nielsen, D. Pisinger, and P. Marquardsen, “Automatic Transformation of Constraint Satisfaction Problems to Integer Linear Form – An Experimental Study,” in *Proceedings of the Workshop on Techniques for Implementing Constraint programming Systems (TRICS)*, 2000.
- [8] S. D. Prestwich, S. A. Tarim, and B. Hnich, “Template Design Under Demand Uncertainty by Integer Linear Local Search,” *International Journal of Production Research*, vol. 44, no. 22, pp. 4915–4928, 2006.
- [9] S. A. Tarim and I. Miguel, “A Hybrid Benders’ Decomposition Method for Solving Stochastic Constraint Programs with Linear Recourse,” in *Recent Advances in Constraints, Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, ser. LNCS, B. Hnich, M. Carlsson, F. Fages, and F. Rossi, Eds., vol. 3978. Springer, 2005, pp. 133–148.
- [10] S. D. Prestwich and S. Verachi, “Constructive vs Perturbative Local Search for General Integer Linear Programming,” in *Proceedings of the Fifth International Workshop on Local Search Techniques in Constraint Satisfaction (LSCS)*, Sydney, Australia, 2008.
- [11] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*. John Wiley & Sons, 2005.

Preview

This document is a preview version and not necessarily identical with the original.

<http://www.it-weise.de/>

```
@inproceedings{CWL2009TDUEOWMSO,  
  title      = {Template Design using Extremal Optimization with Multiple Search Operators},  
  author     = {Raymond Chiong and Thomas Weise and Bee Theng Lau},  
  booktitle  = {International Conference of Soft Computing and Pattern Recognition (SoCPar 2009)},  
  editor     = {Ajith Abraham and Azah Kamilah Muda and Nanna Suryana Herman and Siti Mariyam Shamsuddin and  
              Choo Yun Huoy},  
  location  = {Malacca, Malaysia},  
  month     = dec # {,~4--7},  
  year      = {2009},  
  url       = {http://www.it-weise.de/documents/index.html#CWL2009TDUEOWMSO},  
  url       = {ftp://ftp.computer.org/press/outgoing/proceedings/Patrick/socpar09/data/3879a202.pdf},  
  publisher = {Conference Publishing Service (CPS)},  
  address   = {},  
}
```