A Novel Memetic Algorithm with Explicit Control of Diversity for the Menu Planning Problem

Carlos Segura*, Gara Miranda†, Eduardo Segredo†, Joel Chacón*

*Área de Computación
Centro de Investigación en Matemáticas, A.C.
Guanajuato, Mexico
†Dpto. de Ingeniería Informática y de Sistemas
Universidad de La Laguna, Spain
Email: carlos.segura@cimat.mx, gmiranda@ull.edu.es, esegredo@ull.edu.es, joel.chacon@cimat.mx

***Abstract*—Menu planning is a complex task that involves finding a combination of menu items by taking into account several kinds of features, such as nutritional and economical, among others. In order to deal with the menu planning as an optimization problem, these features are transformed into constraints and objectives. Several variants of this problem have been defined and metaheuristics have been significantly successful solving them. In the last years, Memetic Algorithms (MAs) with explicit control of diversity have lead the attainment of high-quality solutions in several combinatorial problems. The main aim of this paper is to show that these types of methods are also viable for the menu planning problem. Specifically, a simple problem formulation based on transforming the menu planning into a single-objective constrained optimization problem is used. An MA that incorporates the use of iterated local search and a novel crossover operator is designed. The importance of incorporating an explicit control of diversity is studied. This is performed by using several well-known strategies to control the diversity, as well as a recently devised proposal. Results show that, for solving this problem in a robust way, the incorporation of explicit control of diversity and ad-hoc operators is mandatory.**

***Index Terms*—Menu Planning, Diversity, Memetic Algorithm**

I. Introduction

According to the *World Health Organization*, nutrition is one of the main pillars of health and development. In general, healthy people are stronger, more productive and better able to break the cycle of poverty and hunger and to develop their full potential. Malnutrition carries considerable risks to human health. Currently, the world faces a double burden of malnutrition that includes both malnutrition and over-nutrition or overweight. The increase in overweight and obesity throughout the world is one of the main challenges for public health. People of all ages and conditions face this type of malnutrition, as a result of which the rates of diabetes, cardiovascular diseases and other diseases related to the diet are increasing dramatically. When the above problems arise in children, the situation is even more delicate. In any case, the only way to avoid this situation is to follow a healthy and balanced diet.

Controlling the feeding of children at home is quite an impossible task for public health organizations. However, educational institutions can take steps to improve the nutrition of children at schools. In most schools, students are offered at least one meal per day (usually lunch). The set of dishes or food offered in school cafeterias should be proposed and periodically reviewed by an expert in order to verify that school menus fulfill healthy and appropriate nutritional recommendations. In practice, most healthy and balanced diets are manually designed by a nutrition expert. An expert trained professionally in the field of nutrition or food science is able to define specific meals that are ideally suited to a person (or group of people) nutritional requirements. However, by considering general criteria for children nutrition and a predefined set of meal choices, it is possible to properly formulate the problem of automatically generating healthy and balanced plans for school menus. Such a problem is usually denoted in the literature as *Menu Planning Problem* (MPP) [1].

In this work, we propose the use of a novel Memetic Algorithm (MA) with explicit control of diversity in order to address the MPP. The reason to include explicit control of diversity is that, in the last years, results of several well-known combinatorial optimization problems have been improved further by incluiding these control mechanisms [2]. Most of such methods are usually applied with large populations, where they have shown to be successful in the very long term, i.e., by considering executions of several days or even weeks [3]. In this paper, the viability of using these methods, when the computational power is more restricted, is studied. Particularly, executions of just some hours that evolve about 250 generations of small populations are considered. Moreover, a recent strategy that simplifies a well-known but complex strategy is applied. The benefits of incorporating an explicit strategy to control the diversity are clear and the methods that attain better results are those that relate the control of diversity to the stopping criterion. Regarding the design of the MA, another important novelty is the crossover operator. An ad-hoc crossover operator based on preserving the similarities of the parents in the offspring, is designed and compared against the traditional uniform crossover. The incoporation of this operator increases the robustness of the proposal and its advantages appear when dealing with large instances of the MPP.

978-1-7281-2153-6/19/$31.00

II. Menu Planning Problem

A. Literature Review

The MPP [1] involves generating a diet plan, usually daily, weekly or monthly, consisting of healthy menus for the main meals of the day. The different meals that comprise every menu in the plan should be fully specified. For example, for lunch, we could specify the starter, the main course, the dessert, the drink and the side dishes. The overall plan must fulfill general recommendations for healthy and balanced diets. Such recommendations must consider human energy requirements depending on physiological features of the person (e.g., age, gender, height) and lifestyle (e.g., amount and frequency of physical exercise) in order to establish an adequate proportion for macronutrients (carbohydrates, proteins and fats) and micronutrients (vitamins and minerals) intakes. User data and personal preferences can be used to design completely specific meal plans [4]–[6]. However, that information does not make sense when planning menus intended for groups of people, which is the goal herein.

The most important and common consideration for menu planning problems, in general, is to ensure a healthy and balanced menu from a nutritional point of view. These nutritional requirements are usually grouped into multiple constraints, since they must provide the recommended minimum and maximum amounts of different nutrients [5], [7]–[11]. Other issues—as the variety, quality (seasonal, fresh, etc.) or cooking method of the food—also impact on the healthiness of the diet. Studies have shown that food variety and/or diversity are healthy and related to diet quality [12]–[15]. The variety of nutrients is obtained by eating all kind of foods, so all food groups must be considered (cereals, fruits, vegetables, meats, etc.), since no single food contains all the necessary nutrients by itself. However, some food groups provide more nutrients than others so it's important to eat everything, but also in the right proportions.

Within the variety of foods that make up a menu, we must consider that some people have food intolerances or allergies (cereals, nuts, legumes, seafood, fish, eggs, cow's milk), illnesses or food incompatibilities (celiac disease, diabetes, semi-vegetarianism, vegetarianism, veganism) that prevent the intake of certain foods. Such situations should be managed as problem constraints since the intake of these foods must be completely avoided. Unfortunately, not many papers in the related literature deal with possible user intolerances or food incompatibilities [4], [6].

Some other interesting considerations in the problem field are related to: personal preferences on the type of food, cost of the menus, aesthetic characteristics of the food, preparation time, among others [1]. These extra considerations do not influence the menu quality or how healthy the menu is, so they are usually considered as possible optimization objectives. The issue of what aspects to consider as problem objectives and what to consider as constraints may differ among authors. However, there is a clear tendency to consider the cost of the menu as an objective to be minimised [8]–[10], [16]. Other authors—who do not take into account the cost of the menu—usually include the user preferences for certain foods, the level of adequacy or the level of acceptance as the objective to be optimised [4], [6], [11], [17].

B. Problem Formulation

The main goal herein is to automatically design meal plans for school cafeterias; therefore, only lunch menus are considered through the plan, which excludes all other daily meals. Moreover, the plans are designed for large groups of children, so specific user preferences or food incompatibilities are not considered for the automatic generation of menus. Nevertheless, from the obtained plans we would be able to apply a minimum set of changes in order to avoid possible allergies, intolerances or food incompatibilities.

A solution consists of a meal plan in which there are as many meals—consisting of a starter, main course and dessert—as there are days for which the plan is going to be designed. We should note that in an effort to design appropriate meal plans, it was necessary to design a database of courses and ingredients with information on their respective nutritional specifications and nutritional advice, as well as on their allergens. To this end, we have taken into account the *Spanish Database of Food Composition*[1], developed in conjunction with the Spanish Agency for Food Safety and Nutrition and endorsed by the Ministry of Health, Social Services and Equality and the Ministry of Science and Innovation of the Government of Spain. This database contains, in a very complete way, nutritional information of hundreds of foods per 100 grams of edible product. We also consulted several consensus documents and guides on school diets and allergens endorsed by the Spanish Government *Ministry of Education, Culture and Sports* and the *Ministry of Health, Social Services and Equality*[2].

The nutritional quality of the meal plans are treated as the problem constraints. Usually, only the total intakes for energy, fats, carbohydrates and proteins are considered when generating meal plans. However, the presence of other nutrients in the diet—as vitamins or minerals—is essential to improve immunity and foster a healthy development. General recommendations on intakes of nutrients—given in the *White Book on Child Nutrition*[3]—are considered in this formulation of the MPP. The recommended daily amount of nutrients for children between 4 and 13 years old does not differ significantly by gender, and in practice, such a distinction is not made in school cafeterias.

For each meal plan generated, all of its nutrients will be calculated. For each of the nutrients, its value has to be within the acceptable range of its recommendation; otherwise, the solution will be considered non-feasible. At this point, we should note that, since we are only considering lunch in the meal plan, recommendation ranges apply to quantities to be

[1] http://www.bedca.net/

[2] http://www.aecosan.msssi.gob.es and https://www.aepnaa.org/

[3] https://www.aeped.es/comite-nutricion/documentos/libro-blanco-nutricion-infantil

Nutrient	r_{min}	r_{max}
Energy (per day)	0.85	1.15
Fats (per day)	0.75	1.25
Proteins (per day)	0.75	1.25
Energy (n days)	0.90	1.10
Fats (n days)	0.90	1.10
Proteins (n days)	0.90	1.10
Folic acid	0.70	1.30
Phosphorus	0.70	1.30
Magnesium	0.70	1.30
Selenium	0.70	1.30
Sodium	0.70	1.30
Vitamin A	0.70	1.30
Vitamin B1	0.70	1.30
Vitamin B2	0.70	1.30
Vitamin B6	0.70	1.30
Vitamin B12	0.70	1.30
Vitamin C	0.70	1.30
Vitamin D	0.70	1.30
Vitamin E	0.70	1.30
Iodine	0.70	1.30
Zinc	0.70	1.30

TABLE I
MINIMUM AND MAXIMUM RANGES FOR NUTRIENTS INTAKE

proportionally ingested at lunch time. The amount of nutrient h to be ingested every day at lunch time is denoted as (r_h). There is therefore a set R of pairs (r_{min}, r_{max}) and size $|H|$, where elements r_{min_h} and r_{max_h} represent the minimum and maximum proportion of nutrient h allowed within the global plan for n days. The set of nutrients and corresponding ranges considered in this work are presented in Table I. Formally, a feasible individual S—from a nutritional point of view—should fulfil the following constraints:

$$\forall h \in H : n \times r_{min_h} \times r_h \leq in(S,h) \leq n \times r_{max_h} \times r_h \quad (1)$$

where $in(S,h)$ denotes the global intake of element h in the whole menu. Only for energy, fats and proteins — the three first elements in Table I — the intakes for every single day d, $in(S,h,d)$, are also checked to be in the established diary ranges ($n = 1$).

As described, some constraints are considered per day and some other ones are considered globally for the whole menu. Many optimizers require the definition of an infeasibility degree (id) to compare solutions, which usually considers the relative importance of the constraints. Lets denote by HD the elements that are considered for the daily constraints (they are the three first elements of Table I) and HG the ones that are considered globally (all the remaining ones). Then, the infeasibility degree $id(S)$ is calculated as follows:

$$id(S) = did(S) + gid(S) \quad (2)$$

$$did(S) = \sum_{h \in HD} \sum_{d=1}^{n} (max(in(S,h,d), r_{max_h}) - r_{max_h})^2 + \sum_{h \in HD} \sum_{d=1}^{n} (r_{min_h} - min(in(S,h,d), r_{min_h}))^2 \quad (3)$$

$$gid(S) = \sum_{h \in HG} (max(in(S,h), r_{max_h}) - r_{max_h})^2 \times 10^6 + \sum_{h \in HG} (r_{min_h} - min(in(S,h), r_{min_h}))^2 \times 10^6 \quad (4)$$

In Equation (2), $did(S)$ quantifies the infeasibility of daily constraints. Note that distances to the acceptable ranges are squared with the aim of heavily penalizing the constraints that are strongly unfulfilled. Finally, $dig(S)$ quantifies the infeasibility of global constraints. As in previous case, distances are squared. Addionally, since global constraints are considered to be more important, the result is multiplied by 10^6.

Finally, we need to define the optimization goal for the MPP. In school cafeterias, meal prices should be reasonable, so we have fixed the menu price as a goal to be minimised. In fact, cost is one of the most common optimization objectives regarding the MPP literature. We should note at this point that the total cost of a meal plan is calculated as the sum of the costs of all its meals. Moreover, the cost of a meal is calculated as the sum of the costs of all the ingredients—considering their respective amounts—required to prepare that meal.

The total cost of a meal plan for n days is calculated as follows:

$$C = \sum_{j=1}^{n} c_{st_j} + c_{mc_j} + c_{ds_j} \quad (5)$$

where c_{st_j}, c_{mc_j} and c_{ds_j} are the costs for the starter, main course and dessert, respectively, for day $j \in \{1, \ldots, n\}$.

III. EXPLICIT CONTROL OF DIVERSITY IN EVOLUTIONARY ALGORITHMS

Most variants of Evolutionary Algorithms (EAS) start with a set of diverse candidate solutions, and as the generations evolve, such a diversity is reduced [18]. When all the population members are similar to each other, it is said that the population have converged. In the convergence status, the typical operators of EAs are not able to improve further such solutions so in most cases, the computational resources are wasted until the execution finishes. Premature convergence is one of the most common drawbacks of EAs [19], so many techniques for dealing with it have been devised [20]. The methods are classified, depending on the component of the EA that is modified, into the following groups [18]: *selection-based*, *population-based*, *crossover/mutation-based*, *fitness-based*, and *replacement-based*.

In recent years, replacement-based strategies have attained impressive results, so this paper focuses on such kind of strategies. One of the first techniques categorized as replacement-based are the *crowding* proposals. In crowding, the basic principle is to erase individuals from the population that are similar to those offspring that are selected to survive. This can be accomplished in many different ways. One of the most popular realization is the *Restricted Tournament Selection* [21] (RTS) strategy. In RTS after a new individual (C) is created, CF individuals from the current population are randomly

Algorithm 1 Memetic Algorithm for the Menu Planning — MAMP

1: **Initialization**: Generate an initial population P_0 with N random individuals. Assign $t = 0$.
2: **Iterated Local Search**: Perform an Iterated Local Search with stopping criterion set to 100 iterations, for every individual in the population.
3: **while** (not stopping criterion) **do**
4: **Mating selection**: Perform binary tournament selection on P_t in order to fill the mating pool.
5: **Variation**: Apply crossover operator to the mating pool to create a child population CP.
6: **Iterated Local Search**: Perform an Iterated Local Search with stopping criterion set to 100 iterations, for every individual in CP.
7: **Survivor selection**: Apply a survivor selection strategy taking into account P_t and CP.
8: $t = t + 1$
9: **end while**

selected. Then, C and its most similar individual—from those in the selected set—compete for a place in the population using a traditional binary tournament, i.e. the best one survives. The creation of individuals is done as in steady-state schemes.

In some methods, diversity is considered to calculate a fitness value for the replacement stage. For instance, in the *Hybrid Genetic Search with Adaptive Diversity Control* (HGSADC) [22], the individuals—union of parents and offspring—are sorted by their contribution to diversity and by their original cost. In order to calculate the contribution to diversity, the mean distance to the closest N_{Close} individuals is calculated. For an individual I, the ranking in terms of diversity is denoted as $RD(I)$ whereas the ranking for the original cost is denoted as $RC(I)$. Then, the rankings are combined to generate the biased fitness value $BF(I)$ using Equation (6). Note that this scheme requires the setting of two parameters: N_{Close} and N_{Elite}, whereas N_{Indiv} refers to the number of individuals that has not been erased yet by the replacement scheme. In each step of the replacement phase, the individual with the lowest fitness is erased and the ranks are recalculated. This is iteratively performed until the desired population size is attained.

$$BF(I) = RC(I) + (1 - \frac{N_{Elite}}{N_{Indiv}}) \times RD(I) \qquad (6)$$

The *contribution-diversity replace-worst* method [23] (CDRW) is another steady-state approach. After the creation of an individual, it enters in the population by replacing another one that is worse both in diversity contribution and quality. If such an individual does not exist, the *replace-worst* strategy (RW) is applied, i.e. the worst individual taking only into account its quality is erased.

Finally, using concepts that arise in the field of multi-objective optimization to consider both the original objective and the diversity contribution of each individual is also a plausible choice [24]. A recent variant that uses this principle is the *replacement with multiobjective based dynamic diversity control strategy* (RMDDC). One of the main differences between this proposal and the previous ones is that it applies the principle of adapting the degree of exploration and exploitation to the requirements of the different optimization stages by taking into account the stopping criterion set by the user. Particularly, a minimum desired contribution to diversity (D) is calculated dinamically. Specifically, the initial D (D_I) is set by calculating the mean distance between individuals in the first generation and multiplying it by an initial distance factor (IDF). Then, D is reduced linearly, so that at the end of the execution it is equal to 0. The replacement phase is based on an iterative process to select all the survivors. In each iteration, among the individuals with a diversity contribution larger than D, one is selected by considering both its contribution to diversity and its quality. Particularly, these two values are considered as the two objectives to optimize and the non-dominated set is calculated. Then, an individual of the non-dominated set is selected randomly. For additional details and pseudocode, readers are referred to [25].

Algorithm 2 Uniform Crossover — UX

Require: Parent1, Parent2
1: Child1 = Parent1
2: Child2 = Parent2
3: **for** i in $\{0, 1, 2, ..., 3 \times n - 1\}$ **do**
4: **if** (Random([0,1]) $\leq$ 0.5) **then**
5: Swap(Child1[i], Child2[i])
6: **end if**
7: **end for**

IV. Algorithmic Proposals

Our general proposal (Memetic Algorithm for Menu Planning — MAMP) is a first generation MA that applies Iterated Local Search (ILS) as the improvement strategy. The general algorithm is a quite standard Memetic Algorithm (see Algorithm 1). Each solution is encoded using an array with $3 \times n$ elements, where n is the number of days to plan. Positions i that fullfill $i \ mod \ 3 = 0$ refer to the starter at day $\lfloor i/3 \rfloor + 1$. Positions i that fullfill $i \ mod \ 3 = 1$ refer to the main course at day $\lfloor i/3 \rfloor + 1$. Finally, positions i that fullfill $i \ mod \ 3 = 2$ refer to the dessert at day $\lfloor i/3 \rfloor + 1$. The initialization (line 1) is based on random uniform selection of the menu items. The learning strategy (line 2 and line 6) is a standard iterated local search. Parent selection (line 4) is performed with binary tournaments and two different choices were tested for the crossover operator (line 5). Note that in the case of applying a steady-state survivor selection, the CP contains only one individual. Otherwise, it contains N individuals. Finally, six different survivor selection strategies were tested (line 7). In order to fully define our proposal, the details of the crossover operators, iterated local search and survivor selection are required. These are given in the next subsections.

A. Crossover Operators

Two different crossover operators were tested. The first one is the classic uniform crossover (UX) that is illustrated in

Algorithm 3 Similarity Based Crossover — SX

Require: P1, P2 //Parents
1: P1Meals = $\emptyset$ multiset
2: P2Meals = $\emptyset$ multiset
3: **for** i in $\{0, 1, 2, ..., n-1\}$ **do**
4: P1Meals = P1Meals $\cup$ Meal(P1[i*3], P1[i*3+1], P1[i*3+2])
5: P2Meals = P2Meals $\cup$ Meal(P2[i*3], P2[i*3+1], P2[i*3+2])
6: **end for**
7: currentDay = 0
8: **for** meal in (P1Meals $\cap$ P2Meals) **do**
9: Child1[currentDay*3] = meal.starter
10: Child2[currentDay*3] = meal.starter
11: Child1[currentDay*3+1] = meal.main
12: Child2[currentDay*3+1] = meal.main
13: Child1[currentDay*3+2] = meal.dessert
14: Child2[currentDay*3+2] = meal.dessert
15: P1Meals = P1Meals - {meal}
16: P2Meals = P2Meals - {meal}
17: currentDay = currentDay + 1
18: **end for**
19: commonDays = currentDay
20: **for** i in $\{commonDays...., n-1\}$ **do**
21: meal1 = P1Meals.selectRandom()
22: P1Meals = P1Meals - {meal1}
23: meal2 = P2Meals.selectRandom()
24: P2Meals = P2Meals - {meal2}
25: **if** (Random([0,1]) $\leq$ 0.5) **then**
26: Swap(meal1, meal2)
27: **end if**
28: Child1[currentDay*3] = meal1.starter
29: Child2[currentDay*3] = meal2.starter
30: Child1[currentDay*3+1] = meal1.main
31: Child2[currentDay*3+1] = meal2.main
32: Child1[currentDay*3+2] = meal1.dessert
33: Child2[currentDay*3+2] = meal2.dessert
34: currentDay = currentDay + 1
35: **end for**

Algorithm 2. The second operator is called the similarity-based crossover (SX—see Algorithm 3) and it is a newly designed ad-hoc crossover operator. Two principles guided its design. The first one is that since there are some daily contraints in the problem, it make sense to inherit complete meals, i.e. combinations of starter, main and dessert. The second one is that with the formulation given, the constraints and objective function depend on the complete meals that are selected, but not on the order that is considered for them. For instance, given a solution S, interchanging the meals of the first day and second day, does not produce any effect on the calculation of the infeasibility degree and objective function. Thus, candidate solutions might be interpreted as multisets with complete meals. SX identifies the set of meals that are included in both parents and their corresponding cardinalities (lines 1 to 8). Then, each common meal is directly copied to both children (lines 8 to 18). Finally, for the remaining meals, which appear only in one of the parents, a uniform crossover is used. Specifically, meals are extracted in random order (lines 21 and 24) and with a 0.5 probability they are interchanged prior to including them in the children. Thus, the main aim of this crososver is to first identify the similarities among parents, and then preserve such similarities in the children.

Algorithm 4 Iterated Local Search — ILS

Require: S (Initial Solution), Iterations
1: Best = S
2: **for** i in $\{1, 2, ..., Iterations\}$ **do**
3: S' = First-Improvement-Hill-Climbing(S)
4: **if** (S' is Better than Best) **then**
5: Best = S'
6: **end if**
7: S = Perturb(Best)
8: **end for**
9: **return** Best

B. Iterated Local Search

Our proposal uses a quite simple variant of ILS (see Algorithm 4), where in each iteration a local search is applied (line 3). The local search procedure is the well-known first-improvement hill climbing. Each neighbor is generated by altering a single menu item. Thus, the number of neighbours of any solution is $n \times (STO + MCO + DSO - 3)$, where STO, MCO and DSO are the number of options for starters, main courses and desserts, respectively. Neighbours are explored in a random way, and any modification that improves the current solution is accepted. The first-improvement hill climbing stops when a local optimum is reached. Note that, as usual, since a constrained optimization problem is used the way to compare solutions is the following. Given two solutions $S1$ and $S2$, if their infeasibility degree are different, the one with lower infeasibility degree is better. Otherwise, if their infeasibility degrees are equal, then the one with better objective—minimum cost in this case—is better.

The perturbation (line 7) is always applied to the best solution found so far and it operates as follows. In the cases where the solution to perturb contains some days with contraints unfulfilled, all these days are perturbed by reassigning its three meal items randomly. If all the daily contrainsts are fulfilled, but some global constraints are unfulfilled, the perturbation operates as follows. First, the macronutrient or micronutrient element that contributes higher to the infeasibility degree is detected. If for this element the consumption is higher than the desired, one of the six days with a higher consumption of such element is selected randomly and perturbed by reassigning its three meal items randomly. Similarly, if the consumption is lower than the desired, one of the six days with a lower consumption of such element is selected randomly and perturbed by reassining its three meal items randomly. Finally, for cases in which all the constraints are fulfilled, a single day is selected at random and it is perturbed by reassigning its three meal items randomly.

C. Survivor Selection Strategies

Six different survivor selection strategies have been used in this paper. The first five strategies have been used extensively in several papers so pseudocodes are not included. The first one is the generational with elitism (GEN_ELIT) strategy. In this case, the whole offspring survives. Additionally if the best solution of the previous population is better than any

Algorithm 5 BNP survivor selection technique

Require: Population, Offspring
1: Penalized = $\emptyset$ multiset
2: CurrentIndividuals = Population $\cup$ Offspring
3: Best = Best Individual in CurrentIndividuals
4: NewPop = { Best }
5: CurrentIndividuals = CurrentIndividuals - { Best }
6: Update D taking into account the elapsed time (T_e), stopping criterion (T_s) and initial value of D (D_I): $D = D_I - D_I * \frac{T_e}{T_s}$. We took $D_I = distInitFactor * M$, where M is the mean distance in the first population and $distInitFactor = 0.5$.
7: **while** (|NewPop| < N) **do**
8: **for all** I $\in$ CurrentIndividuals **do**
9: I.DCN = distance to the closest individual of I in NewPop
10: **if** I.DCN $< D$ **then**
11: Penalized = Penalized $\cup$ {I}
12: CurrentIndividuals = CurrentIndividuals - {I}
13: **end if**
14: **end for**
15: **for all** I $\in$ Penalized **do**
16: I.DCN = distance to the closest individual of I in NewPop
17: **end for**
18: **if** CurrentIndividuals is empty **then**
19: Selected = I with largest I.DCN in Penalized
20: Penalized = Penalized - {Selected}
21: **else**
22: Selected = Best Individual in CurrentIndividuals
23: CurrentIndividuals = CurrentIndividuals - {Selected}
24: **end if**
25: NewPop = NewPop $\cup$ {Selected}
26: **end while**
27: Population = NewPop

TABLE II
PARAMETERIZATION OF THE SURVIVOR SELECTION STRATEGIES

Survivor Selection	Parameterization
BNP	$IDF = 0.5$
RMDDC	$IDF = 0.5$
HGSADC	$N_{Close} = 3$, $N_{Elit} = 3$
RTS	$CF = 5$
CDRW	No parameters required
GEN_ELIT	No parameters required

of the offspring, it survives. Four of the additional strategies applied were already introduced in this paper. They are the RTS, HGSADC, CDRW and RMDDC strategies. Finally, a simplification of the RMDDC, which is called the Best Non-Penalized Survivor Strategy (BNP—see Algorithm 5) is used. Since this is a recently proposed strategy [26], all the details are specified.

One of the principles of the BNP strategy is to avoid the selection of too close individuals. Specifically, the approach tries to avoid the selection of pairs of individuals that are closer than a D value. Since in the initial phases it is important to explore, whereas in the last phases the procedure should intensify, the D value varies during the execution. Particularly, the variable D is initialized in base of the content of the first population (see line 6) and it is decreased in a linear way.

In each execution of the survivor operator, N individuals from the previous population and offspring are selected to survive. BNP iteratively selects the best element that is not penalized, i.e. with a distance larger than D to the already selected individuals. Then, all the remaining individuals with distance at most D to the previously selected ones are penalized. If it is not possible to find a non-penalized individual, the individual with larger distance to the currently selected individuals is taken. Note that the principles of BNP are similar to the ones that guided the design of the RMDDC [25] strategy. The main difference is that in RMDDC the same importance is given to quality and diversity, so a multi-objective selection is used, which increase the complexity (both in terms of understanding and computational steps) of the approach.

The survivor operator requires a distance-like function between individuals. Given two individuals $S1$ and $S2$, a multiset with the meals taken can be generated with lines 3 to 6 of Algorithm 3. Lets denote these two multisets as S1Meals and S2Meals. The cardinality of the intersection of these two multisets measures the similarity between $S1$ and $S2$. Thus, Equation (7) is used to measure the distance between them.

$$Dist(S1, S2) = n - |S1Meals \cap S2Meals| \quad (7)$$

V. EXPERIMENTAL ASSESSMENT

This section is devoted to present the experimental validation of the different proposals described in this paper. Results for three different cases are presented. The same database and nutrition recommendations are used in all of them as previously described. Thus, the only difference is the number of days to plan. Particularly, $n = 20, 40, 60$ were used. The database contains 18 starters, 33 main courses and 13 desserts, meaning that the search space size is about 10^{77}, 10^{155} and 10^{233}, respectively.

Since all the proposals are stochastic, each execution was repeated 30 times. All experiments were performed by setting a common stopping criterion for all the algorithms. In the case of using $n = 20$, the stopping criterion was set to 1 hour, whereas it was increased to 2.5 and 5 hours, for the cases $n = 40$ and $n = 60$, respectively. All the proposals were able to generate feasible solutions in all the executions. Thus, comparissons are performed in base of the attained cost. In order to compare the different methods, the series of statistical tests described in [27] were used. This guideline takes into account the ANOVA, *Shapiro-Wilk*, *Levene*, *Welch* and *Kruskal-Wallis* tests. They were applied assuming a significance level of 5%.

Note that six different survivor selection strategies and two different crossover operators were implemented. Thus, the study involves the application of twelve algorithms. In order to refer to each method, a label with two parts separated with a dash is used. The first part denotes the survivor selection strategy, whereas the second part denotes the crossover strategy. Some of the survivor selection strategies require additional parameters. They were set by considering some preliminary experiments and the final applied parameterization is shown in Table II. The only additional parameter, which was required by all proposals, is the population size. It was set to 10

TABLE III
SUMMARY OF THE COST ATTAINED BY THE DIFFERENT PROPOSALS

	20 Days			40 Days			60 Days		
	Best	**Worst**	**Mean**	**Best**	**Worst**	**Mean**	**Best**	**Worst**	**Mean**
BNP-SX	20.265	20.265	20.265	40.038	40.067	40.059	59.721	59.788	59.733
BNP-UX	20.265	20.265	20.265	40.038	40.282	40.072	59.721	60.189	59.832
RMDDC-SX	20.265	20.265	20.265	40.038	40.282	40.068	59.721	59.941	59.738
RMDDC-UX	20.265	20.265	20.265	40.038	40.427	40.098	59.721	60.512	60.118
HGSADC-SX	20.265	20.265	20.265	40.038	40.339	40.088	59.721	59.942	59.740
HGSADC-UX	20.265	20.265	20.265	40.038	40.282	40.080	59.721	60.370	59.846
RTS-SX	20.265	20.265	20.265	40.038	40.339	40.151	59.721	59.960	59.784
RTS-UX	20.265	20.265	20.265	40.067	40.427	40.154	59.721	60.370	59.873
CDRW-SX	20.265	20.460	20.297	40.067	40.434	40.260	59.721	60.543	59.813
CDRW-UX	20.265	20.460	20.271	40.067	40.409	40.183	59.721	60.189	59.852
GEN_ELIT-SX	20.265	20.469	20.297	40.067	40.434	40.249	59.721	59.942	59.779
GEN_ELIT-UX	20.265	20.460	20.290	40.038	40.434	40.233	59.721	60.105	59.843

TABLE IV
STATISTICAL COMPARISSON OF THE DIFFERENT PROPOSALS FOR THE THREE INSTANCES

	20 Days				40 Days				60 Days			
	↑	↓	↔	**Score**	↑	↓	↔	**Score**	↑	↓	↔	**Score**
BNP-SX	3	0	8	3	10	0	1	10	9	0	2	9
BNP-UX	3	0	8	3	6	1	4	5	1	3	7	-2
RMDDC-SX	3	0	8	3	8	0	3	8	9	0	2	9
RMDDC-UX	3	0	8	3	4	2	5	2	0	11	0	-11
HGSADC-SX	3	0	8	3	5	1	5	4	9	0	2	9
HGSADC-UX	3	0	8	3	5	2	4	3	1	3	7	-2
RTS-SX	3	0	8	3	3	3	5	0	1	3	7	-2
RTS-UX	3	0	8	3	2	5	4	-3	1	4	6	-3
CDRW-SX	0	8	3	-8	0	8	3	-8	1	3	7	-2
CDRW-UX	0	0	11	0	0	6	5	-6	1	3	7	-2
GEN_ELIT-SX	0	8	3	-8	0	8	3	-8	2	3	6	-1
GEN_ELIT-UX	0	8	3	-8	0	7	4	-7	1	3	7	-2

individuals with the aim of reducing the time required by the executions. Preliminary experiments showed that higher values attain similar results but require additional time.

Table III shows a summary of the results attained by each proposal. Particularly, the best (minimum), worst (maximum) and mean of the cost is shown for the three different instances and for the twelve algorithms. In order to facilitate the analysis, pair-wise statistical analyses among all the tested algorithms were performed using the set of statistical tests previously mentioned. Table IV shows a summary of the results of the statistical tests. Columns labelled with a $\uparrow$ show the amount of times that each algorithm was the winner in the pair-wise statistical tests. The times that each algorithm loses is shown in the columns labelled with a $\downarrow$, whereas the columns tagged with a $\leftrightarrow$ refer to the amount of times where differences were not stastically significant. Finally, the score is calculated as the times than an algorithm wins minus the times than an algorithm loses, and it gives an overall performance measure of each proposal.

Attending to the results attained in the "20 Days" instance, several of the algorithms are similar. In fact, 8 algorithms attained the same result in all the executions. The proposals that used the CDRW or GEN_ELIT as replacement strategy were the only ones that reported some no so high-quality results in some executions. The overall score shows a tie among the remaining eight algorithms. This means that for small cases using an ad-hoc crossover operator is not really important for the performance.

In the case of the "40 Days" instance, larger differences appear among the proposals. In this case, the BNP-SX strategy is the one that attains the lowest mean. Moreover, statistical tests report that it is superior to most of the alternatives. In fact, only when compared to RMDDC-SX, differences were not statistically significant and its score is the highest one. It is also remarkable that in this case, using an ad-hoc crossover operator such as the SX contributes importantly to the performance. The sum of the scores of the methods that apply SX is 6, whereas the sum of the methods that apply UX is -6. Thus, for attaining proper solutions it is important to select both a proper replacement strategy and crossover operator.

Finally, in the case of the "60 Days" instance, the BNP-SX strategy is again the one that attains the lowest mean. However, in this case, statistical tests show that RMDDC-SX and HGSADC-SX attain the same score. In this case, differences in the performance among those proposals that apply SX and those that apply UX are clearer. Particularly, the sum of the scores of the methods that apply SX is 22 and for those that apply UX is -22. Thus, in order to attain scalable algorithms, ad-hoc crossover operators are required. Again for this instance, results show that both a proper replacement and

an ad-hoc operator are required to develop robust optimizers capable of attaining high-quality solutions in most of the executions.

Taking into account all the results, the BNP-SX and RMDDC-SX strategies were the clear winners, with a slight superiority for the BNP-SX strategy. The common feature of BNP and RMDDC is that they relate the amount of diversity maintained in the population to the stopping criterion. In fact, they are the only methods that start with a high diversity in the population and decrease it gradually during the whole execution.

VI. Conclusions and further research

In the current work, we have proposed a MA, which integrates a novel ad-hoc crossover operator, as well as a novel diversity control mechanism in the form of a replacement scheme, to deal with the MPP. In order to analyse the contribution of both, the novel crossover operator and the replacement scheme, the well-known uniform crossover, and additional replacement schemes were also embedded into the MA.

The experimental evaluation performed demonstrated that incorporating explicit diversity control mechanisms into the MA was able to improve its robustness, as well as its results, in terms of the quality of the solutions attained at the end of the runs, particularly in the case of larger instances. We note that, in the case of the MPP, those diversity control schemes promoting exploration-exploitation depending on the stopping criterion set by the user, were able to provide the best performance. Finally, it was not only important to include a proper diversity control mechanism to improve the performance of the MA, but also to consider a tailored crossover operator.

Acknowledgment

This work was supported by the Spanish Ministry of Economy, Industry and Competitiveness as part of the programme "I+D+i Orientada a los Retos de la Sociedad", with contract number TIN2016-78410-R. Authors acknowledge the financial support from CONACyT through the "Cienca Básica" project no. 285599.

References

[1] H. Ngo, Y. Cheah, O. Goh, Y. Choo, H. Basiron, and Y. J. Kumar, "A review on automated menu planning approaches," *Journal of Computer Science*, vol. 12, pp. 582–596, 01 2016.

[2] C. Segura, A. H. Aguirre, S. I. V. Peña, and S. B. Rionda, *The Importance of Proper Diversity Management in Evolutionary Algorithms for Combinatorial Optimization*. Cham: Springer International Publishing, 2017, pp. 121–148.

[3] C. Segura, S. Botello Rionda, A. Hernández Aguirre, and S. I. Valdez Peña, "A novel diversity-based evolutionary algorithm for the traveling salesman problem," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 489–496.

[4] O. Chávez, J. Marchi, and P. Pozos, "Nutritional menu planning: A hybrid approach and preliminary tests," *Research in Computing Science*, vol. 82, pp. 93–104, 2014.

[5] B. Hernández-Ocaña, O. Chávez-Bosquez, J. Hernández-Torruco, J. Canul-Reich, and P. Pozos-Parra, "Bacterial foraging optimization algorithm for menu planning," *IEEE Access*, vol. 6, pp. 8619–8629, 2018.

[6] T. Isokawa and N. Matsui, "Performances in ga-based menu production for hospital meals," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 2498–2501.

[7] J. Aberg, "An evaluation of a meal planning system: Ease of use and perceived usefulness," in *Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology*, ser. BCS-HCI '09. Swinton, UK, UK: British Computer Society, 2009, pp. 278–287.

[8] S. Gumustekin, T. Senel, and M. Cengiz, "A comparative study on bayesian optimization algorithm for nutrition problem," *Journal of Food and Nutrition Research*, vol. 2, no. 12, pp. 952–958, 2014.

[9] J. Hsiao and H. Chang, "Smartdiet: A personal diet consultant for healthy meal planning," in *2010 IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS)*, Oct 2010, pp. 421–425.

[10] A. Kahraman and H. Seven, "Healthy daily meal planner," in *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 390–393.

[11] T. Kashima, S. Matsumoto, and H. Ishii, "Evaluation of menu planning capability based on multi-dimensional 0/1 knapsack problem of nutritional management system," *IAENG International Journal of Applied Mathematics*, vol. 39, no. 3, 2009.

[12] N. Steyn, J. Nel, G. Nantel, G. Kennedy, and D. Labadarios, "Food variety and dietary diversity scores in children: are they good indicators of dietary adequacy?" *Public Health Nutrition*, vol. 9, no. 5, pp. 644–650, 2006.

[13] L. B. Dixon, F. J. Cronin, and S. M. Krebs-Smith, "Let the pyramid guide your food choices: Capturing the total diet concept," *The Journal of Nutrition*, vol. 131, no. 2, pp. 461S–472S, 2001.

[14] A. Drewnowski, S. A. Renderson, A. Driscoll, and B. J. Rolls, "The dietary variety score: Assessing diet quality in healthy young and older adults," *Journal of the American Dietetic Association*, vol. 97, no. 3, pp. 266 – 271, 1997.

[15] A. Kant, A. Schatzkin, T. Harris, R. Ziegler, and G. Block, "Dietary diversity and subsequent mortality in the first national health and nutrition examination survey epidemiologic follow-up study," *The American Journal of Clinical Nutrition*, vol. 57, no. 3, pp. 434–440, 1993.

[16] D. Osthus, "A genetic algorithm approach to optimize planning of food fortification," Ph.D. dissertation, Department of Statistics, ISU, 2011.

[17] N. Funabiki, S. Taniguchi, Y. Matsushima, and T. Nakanishi, "A proposal of a menu planning algorithm for two-phase cooking by busy persons," in *2011 International Conference on Complex, Intelligent, and Software Intensive Systems*, June 2011, pp. 668–673.

[18] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 45, no. 3, pp. 35:1–35:33, Jul. 2013.

[19] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata." in *Foundations of Genetic Algorithms*, L. D. Whitley, Ed. Morgan Kaufmann, 1992, pp. 187–202.

[20] H. M. Pandey, A. Chaudhary, and D. Mehrotra, "A comparative review of approaches to prevent premature convergence in GA," *Applied Soft Computing*, vol. 24, pp. 1047 – 1077, 2014.

[21] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 24–31.

[22] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows," *Computers & Operations Research*, vol. 40, no. 1, pp. 475 – 489, 2013.

[23] M. Lozano, F. Herrera, and J. R. Cano, "Replacement strategies to preserve useful diversity in steady-state genetic algorithms," *Information Sciences*, vol. 178, no. 23, pp. 4421 – 4433, 2008, including Special Section: Genetic and Evolutionary Computing.

[24] L. T. Bui, H. A. Abbass, and J. Branke, "Multiobjective optimization for dynamic environments," in *2005 IEEE Congress on Evolutionary Computation*, ser. CEC'05, vol. 3, 2005, pp. 2349 – 2356 Vol. 3.

[25] C. Segura, A. Hernández-Aguirre, F. Luna, and E. Alba, "Improving diversity in evolutionary algorithms: New best solutions for frequency assignment," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 539–553, Aug 2017.

[26] E. R. Ruiz and C. Segura, "Memetic algorithm with hungarian matching based crossover and diversity preservation," *Computación y Sistemas*, vol. 22, no. 2, pp. 347–361, 2018.

[27] C. Segura, C. A. C. Coello, E. Segredo, and A. H. Aguirre, "A novel diversity-based replacement strategy for evolutionary algorithms," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 3233–3246, Dec 2016.