1. Introduction

Increasing incidence of chronic diseases (CDs) in modern societies requires a shift from adequate towards optimal nutrition, not only providing us with the energy and nutrients but also contributing to the well-being and health. Optimal nutrition, as part of a healthy lifestyle, is an important factor in the prevention of CDs and may also have a therapeutic potential.

Today, many recommendations and guidelines on optimal nutrition, based on results of advanced research methods and tailored to the needs of a society, are available. They consider the current knowledge of the relationship between our immune system, aetiology of CDs and health status. However, their implementation in practice is difficult for several reasons, including the complexity of dietary recommendations and guidelines and limited health literacy that may lead to misunderstanding. Dollahite et al. (1995) found that professionally designed menus published in diet manuals may fail to meet all recommendations and guidelines.

In this paper, we introduce a computer-based method for menu planning, which applies evolutionary computation. First, we formalize the n-day menu-planning problem, decomposing it into several sub-problems at the daily-menu and meal-planning level. We reduce the problem to a multi-dimensional knapsack problem. Then, we define an evolutionary algorithm that quickly finds a diverse set of feasible solutions (i.e. optimal menus) with the optimum objective functions' values, without examining all the possibilities. As the problem is constrained, infeasible solutions need to be repaired in order to direct the "evolution" towards the feasible regions. We present greedy repairing methods that slightly differ at the global level and the sub-problems' levels. At the meal-planning level, we couple repairing with linear programming to balance infeasible meals. We conclude the paper with the presentation of empirical results, which showed that the evolutionary method may outperform a human. A computer was able to find the Pareto-optimal front of 21-day menus with respect to a dietary advice in equal or less time than a human professional, who designed a daily menu. However, the human factor is still important in the last stage, when a solution has to be selected from the Pareto front.
techniques to build the first computer-based menu planner, which optimized menus for nutritional adequacy and budgeted food cost. Brown (1966) developed primitive techniques for controlling the palatability of individual non-selective menus using random selection techniques. A year later, these techniques were adopted by Eckstein (1967) to satisfy menus with several constraints, comprising: cost, color, texture, shape, calories, variety and acceptability by the target population.

In the 1970s, interest in computer applications in menu planning appeared to wane; lack of funding and inadequate software components were contributing causes.

In the 1990s, when the field of artificial intelligence was revived, the menu-planning problem became popular again. Two types of menu planners, case-based and rule-based, were implemented. One of them was JULIA (Hinrichs, 1992), an interactive menu planner that was used to plan meals to satisfy a group of guests, despite conflicting food preferences and evolving constraints. Ganeshan and Farmer (1995) implemented a Prolog catering system. Marling et al. (1999) designed a menu-planning tool for individuals, taking dietary requirements and personal preferences into account. That tool integrated case-based and rule-based reasoning to meet multiple constraints.

A comprehensive review of the use of optimization techniques based on linear and nonlinear programming was given by Darmon et al. (2002).

2.2. Evolutionary computation

The computer-based method for menu planning we have recently proposed (Koroušić Seljak, 2004) is based on evolutionary computation.

In computer science, evolutionary computation is a subfield of artificial intelligence that involves numerical and combinatorial optimization (CO) problems. Evolutionary computation uses iterative progress, such as growth or development in a population of potential problem solutions. The field comprises many techniques, mostly involving metaheuristic optimization algorithms, such as evolutionary algorithms (EAs) and swarm intelligence (Koroušić and Šilc, 2008). These techniques rely on analogies to natural processes; some of them have been inspired by biological mechanisms of evolution. The first ideas were developed in the 1960s by Holland (1961) and Fogel (Fogel and Owens, 1966), and have already reached a stage of some maturity (Michalewicz, 1996).

In recent years, numerous algorithms taking inspiration from nature have also been proposed to handle continuous optimization problems: real-coded genetic algorithms using some specific operators, evolution strategies using Gaussian mutations with adaptive or self-adaptive update strategies, and differential evolution, to name a few.

As real-world optimization problems may involve objectives, constraints and parameters, which constantly change with time, dynamic consideration using evolutionary computation methods have also raised a lot of interest within the last few years. For these dynamic and uncertain optimization problems the objective of the evolutionary algorithm is no longer to simply locate the global optimum solution, but to continuously track the optimum in dynamic environments, or to find a robust solution that operates optimally in the presence of uncertainties.

Optimization is a procedure of finding and comparing feasible solutions until no better solution can be found. Solutions, which in our case are healthy menus, are termed as follows:

- good or bad in terms of multiple conflicting objectives, such as: cost, quality of ingredients, aesthetic standards or other factors; and
- feasible if they satisfy all the problem constraints that are defined by dietary recommendations and guidelines.

While classical deterministic optimization methods can at best find one solution in one simulation run, evolutionary techniques are more efficient in finding multiple trade-off optimal solutions in a single simulation run. These solutions have a wide range of values for each objective representing the multi-dimensional Pareto-optimal front, requiring an additional decision-making activity for choosing a single solution from the front.

3. Menu-planning model

As today’s computers have few limitations, satisfiable menus can be automatically or semi-automatically generated by efficient software (SW) techniques, but only if the menu-planning process is well defined.

Mathematically, menu planning can be reduced to a multi-dimensional (i.e. multi-constrained and multi-objective) knapsack problem (MDKP), which is a widely studied CO problem that has many direct counterparts (Garey and Johnson, 1979).

3.1. Formulation of the MDKP

Given foods of different values and volumes, the MDKP is to find the most valuable combination of foods that fits in a knapsack of fixed volumes. Values are defined subjectively with respect to food quality, cost and aesthetic parameters (comprising taste, consistency, color, temperature, shape and method of preparation). Knapsack volumes are defined by dietary recommendations and guidelines.

3.2. Complexity of the MDKP

MDKP is easy to formulate, yet its decision problem is NP-complete (Garey and Johnson, 1979). In complexity theory, the NP-complete problems are the most difficult problems, which cannot be solved by exact SW techniques in deterministic

---

A metaheuristic is a heuristic method for solving a very general class of computational problems by combining user-given black-box procedures — usually heuristics themselves — in, it is hoped, an efficient way. The name combines the Greek prefix “meta” (“beyond”, here in the sense of “higher level”) and “heuristic” (from έχος, heuriskein, “to find”).

A metaheuristic is a heuristic method for solving a very general class of computational problems by combining user-given black-box procedures — usually heuristics themselves — in, it is hoped, an efficient way. The name combines the Greek prefix “meta” (“beyond”, here in the sense of “higher level”) and “heuristic” (from έχος, heuriskein, “to find”).
polynomial time but require time that is superpolynomial in the input size.

The knapsack values and the volumes are linear, but highly complex, because they are weakly correlated. As there are at least two optimal solutions that are not indifferent to each other, the problem is multimodal.

Another difficulty is that foods are selected from a food composition database (FCDB), which consists of several thousand items having tens of composition parameters. As a consequence, the decision space contains a large set of potential solutions to the menu-planning problem. Moreover, the problem landscape defined by the decision and the objective space contains several peaks.

However, there exist heuristic SW techniques, such as evolutionary computation techniques, that work “reasonably well” on many problem’s instances, but for which there is no proof that they are both always fast and always produce a good result.

Comprehensive reviews of multi-constrained 0-1 knapsack problems, presenting a subset of MDKPs, and associated heuristic algorithms was given by Chu and Beasley (1998) and by Ishibuchi and Kaige (2003).

4. Evolutionary method for menu planning

We applied the state-of-the-art evolutionary algorithm (EA) NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm) (Deb, 2001) in a multi-level way (Gunawan et al., 2003) to solve the MDKP of menu planning. The main idea behind the method is to develop healthy meals and daily menus independently, guiding the optimization to overall Pareto-optimal n-day menus (Fig. 1). All objectives are treated as equally important. The decision on the best compromise to be chosen among adequate Pareto-optimal solutions is made after the search.

At the meal-planning level we solve three sub-problems (for planning meals that are composed of a breakfast and a morning snack, a lunch and an afternoon snack, and a dinner and a light snack before bedtime, respectively), and at the daily-menu planning level we solve five sub-problems (for daily menus with the main meal consisting of red meat, white meat, fish, soya/legumes, or eggs/curd, respectively).

4.1. Some basic definitions

In general, NSGA-II is a multi-objective EA that can be characterized by the use of three ideas: Pareto-dominance-based “fitness” evaluation, diversity maintenance, and elitism.

The algorithm comprises the following steps:

1. Initialize “population” of potential solutions
2. Evaluate “fitness” of “individuals”
3. Estimate feasibility of individuals
4. Repair infeasible individuals
5. Selection:
   - Non-dominated sorting
   - Individual comparison
6. Recombination: Combine traits of “parents”
7. Mutation: Random walk around an individual
8. Evaluate “offspring” solutions
9. Replacement: Best among parents and offspring (Fig. 2)
10. Return to Step 5 or terminate

The non-dominated sorting procedure identifies the best non-dominated set (i.e. a set of solutions that are not dominated by any individual in the population), discards them from the population temporarily, identifies the next best non-dominated set, and continues till all the solutions are classified.

![Fig. 1. Schematic of the evolutionary method for menu planning. LO – local optimization (greedy repair) and LP – linear programming.](image-url)
4.1.2. Initialization

The "crowding-distance sorting" procedure assigns each potential solution a crowding distance (i.e. an average distance from its nearest neighbours on either side of the solution along each of the objectives of the problem). A particular solution is more crowded than another solution if its front density in the neighborhood is higher.

During selection, NSGA-II uses a crowded comparison operator, which takes into consideration both the non-domination rank of an individual in the population and its crowding distance: i.e. non-dominated solutions are preferred over dominated solutions, but between two solutions with the same non-domination rank, the one that resides in the less crowded region is preferred.

4.1.1. Representation structures

In order to apply NSGA-II to the problem of menu planning, we first encode potential solutions of the n-day menu-planning problem and its sub-problems (of the daily menu and meal planning) by integer-valued coding. In our representation:

- at the n-day menu level, a “chromosome” contains n integer data, \(n \in N, n \geq 2\), carrying the information about n daily menu: \([d_1, d_2, ..., d_n]\);
- at the daily-menu level, a “chromosome” contains m integer data, \(m \in N, m \geq 3\) carrying the information about m basic and not composite meals: \([o_1, o_2, ..., o_m]\);
- at the meal level, a “chromosome” is formed of j pairs \((c_i, x_i)\), where \(c_i\) denotes the FCDB code4 of a food item i and \(x_i\) its quantity expressed in grams: \([c_1, x_1, (c_2, x_2), ..., (c_j, x_j)]\). By default, the number of pairs \(j\) varies between 1 and 10, depending on the number of meal courses.

4.1.2. Initialization

The algorithm starts the “evolution” from an initial “population” of either random potential solutions or solutions (n-day menus) known from experience. The population’s size remains constant over all “generations”.

The menu-planning sub-problems at the daily and the meal level operate on local populations. Initially, the local population of the daily-menu level is filled with decomposed n-day menus from the global population, and the local population at the meal level is filled with decomposed daily menus from the daily-menu population.

Beside the global and the local populations, we use additional pools of potential meal and daily-menu solutions (Fig. 1) that have a function of an archive of the union of solutions generated by each sub-problem. Initially, the pools are empty.

4.1.3. Fitness

Each potential solution is evaluated as good or bad in terms of objectives using its fitness values. These are non-negative integer numbers calculated by the following objective functions:

\[
f_k(x) = \frac{1}{\sum_{i=1}^{n} v_i x_i}, \quad k = 1, 2, f_3(x) = \sum_{i=1}^{N} h_i x_i,
\]

\[
f_4(x) = \sum_{i=1}^{n} \lambda_i f_4,i(x), \quad \lambda_i \geq 0 \land \sum_{i=1}^{n} \lambda_i = 1, \quad f_4,i(x) = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij}(x_i) - \sum_{i=1}^{n} h(x_i)\right) - \sum_{i=1}^{n} h(x_i), \quad (1)
\]

\[
h_{ij}(x_i) = \begin{cases} 0, & \text{if } x_i = 0 \\ 1, & \text{if } x_i > 0 \land v_i = 1 \\ \text{otherwise, } x_i \in N \cup \{0\}, i = 1, 2, ..., n, l = 1, 2, ..., 6, \end{cases}
\]

where \(x_i\) denotes the quantity of the food item \(i\) (expressed in grams), \(v_1\) and \(v_2\) its functionality and quality in the season, respectively, \(\lambda_i\) the scalarization weight associated with the aesthetic parameter \(i\), \(v_{i3}, v_{i4}, v_{i5}, v_{i6}, v_{i7}, v_{i8}\) and \(v_{i9}\) the cost, the taste, the consistency, the color, the temperature, the shape, and the method of preparation, respectively, and \(n_o\) the number of possibilities for the \(l\)th aesthetic parameter. In order to reduce the number of objective functions, we apply the normalized weighted sum scalarization technique for assessment of the meal’s or menu’s variety.

The aim of EA at the global and the local levels is to minimize the objective functions of (1).

4.1.4. Feasibility

We estimate feasibility of a potential solution by constraint functions that differ at the global and the local levels.

At the meal level, the constraints are the least restrictive:

- The energy provided by a meal is limited by a lower and an upper bound:

\[
g_3(x) = \frac{\sum_{i=1}^{N} \omega_{i} x_i}{100} \geq 0.9E, \quad g_4(x) = \frac{\sum_{i=1}^{N} \omega_{i} x_i}{100} \leq 1.1E, \quad (2)
\]

where \(\omega_{i}\) denotes the number of calories in 100 g of the food item \(i\), \(x_i\) the quantity of the item \(i\) expressed in grams, and \(E\) the recommended caloric value for the meal.

- The meal’s energy density is limited both upwards and downwards:

\[
g_5(x) = \frac{\sum_{i=1}^{N} \omega_{i} x_i}{\sum_{i=1}^{N} x_i} \geq 0.5E, \quad g_6(x) = \frac{\sum_{i=1}^{N} \omega_{i} x_i}{\sum_{i=1}^{N} x_i} \geq 1.5E, \quad (3)
\]

Footnote:

4 FCDBs normally consist of composition data for foods ordered by major food groups and subgroups (e.g. plain yogurt made of skim milk and plain yogurt made of whole milk have consecutive food codes).
The lower and the upper bounds for daily sodium intake are set at RDI for the dietary fiber is equal to or less than 10% of RDI/\(C_{15}\) and for saturated fatty acids, trans-fatty acids, water-soluble and fat-soluble vitamins, water, major minerals, and trace minerals, to be termed a feasible solution. Formally, these definitions of constraints are similar to that of Eqs. (5) or (8).

4.1.5. Methods for repairing infeasible individuals

EAs have originally been developed for unconstrained problems. The most common way of incorporating constraints into EAs has been the use of a penalty function. Due to difficulties associated with the penalty methods, we apply a repair method\(^a\) to handle constraints by NSGA-II:

- at the meal level, we first replace each infeasible meal those courses that mostly contribute to the violation of constraints with similar but more appropriate ones (e.g. we replace beef broth with vegetable soup if there is a lack of fiber in a meal), and then convert infeasible solutions into feasible solutions using a deterministic local optimization procedure of linear programming (LP in Fig. 1). This procedure, based on the simplex method (Bhatti, 2000), refines the quantities of foods to satisfy the meal sub-problem constraints.
- at the daily-menu and the n-day menu level, we repair infeasible individuals by replacing critical meals with more appropriate ones. Critical meals are those that do not satisfy the constraints on the major food groups (i.e. breads, cereal, rice, and pasta/vegetables/fruits/milk, yogurt, and cheese/meat, poultry, fish, beans, eggs, and nuts/fats, oils, and sweets). Here, we use the problem-specific knowledge, considering a recommendation (i) a daily menu has to include a variety of foods from each major food group and (ii) an n-day menu has to include a diverse set of foods from the major food groups. There may be limitations on frequency of red meat, fish, potatoes, etc.

For this aim, we apply the Lamarckian repair scheme, in which replacements of critical meals are used to generate new “offspring” (Ishibuchi et al., 2005).

The replacement of meal or menu elements that mostly contribute to the violation of constraints requires a prior sorting of elements. We apply the inverse non-dominated sorting.

\(^a\) We apply the conversion factors 4 for proteins and carbohydrates, and 9 for lipids.

\(^b\) For quantities.

\(^c\) Either on a code or on a quantity.
### Table 2
An optimal menu for patients with chronic kidney disease developed by using the evolutionary algorithm (EA) method (Rotovnik Kozjek et al., 2008).

<table>
<thead>
<tr>
<th>Male chronic patients</th>
<th>Female chronic patients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average body weight/height in Slovenia (kg/cm)</td>
<td>70.2/177</td>
</tr>
<tr>
<td>Recommended caloric value (RCV) (MJ/kcal)</td>
<td>10/2400</td>
</tr>
<tr>
<td>Calculated quantity value (g/ml)</td>
<td></td>
</tr>
<tr>
<td><strong>Female chronic patients</strong></td>
<td><strong>Male chronic patients</strong></td>
</tr>
<tr>
<td><strong>Breakfast (25% of the RCV)</strong></td>
<td></td>
</tr>
<tr>
<td>Low-protein bread &amp; Grilled pepper</td>
<td>175/Prot: 2.9 g; AC: 2.9 g; EAC: 1 g; P: 56.1 mg; K: 106.4 mg</td>
</tr>
<tr>
<td>Olive oil &amp; Blueberries</td>
<td>10/Prot: 0.9 g; AC: 0.3 g; EAC: 0.1 g; P: 56.1 mg; K: 106.4 mg</td>
</tr>
<tr>
<td>Sugar</td>
<td>5/Prot: 0 g; AC: 0 g; EAC: 0 g; P: 0 mg; K: 0 mg</td>
</tr>
<tr>
<td><strong>Morning snack (15% of the RCV)</strong></td>
<td></td>
</tr>
<tr>
<td>Millet, cooked in water, with cream</td>
<td>110/Prot: 3.9 g; AC: 1.5 g; P: 107.7 mg; K: 52.7 mg</td>
</tr>
<tr>
<td>Diet margarine &amp; Stewed pears</td>
<td>10/Prot: 0 g; AC: 0 g; EAC: 0 g; P: 0 mg; K: 0 mg</td>
</tr>
<tr>
<td>Stewed pears &amp; Stewed rice</td>
<td>240/Prot: 10.6 g; AC: 0.1 g; P: 132.1 mg; K: 467.2 mg</td>
</tr>
<tr>
<td>Cucumber salad</td>
<td>190/Prot: 1 g; AC: 0.3 g; P: 26.1 mg; K: 286.9 mg</td>
</tr>
<tr>
<td><strong>Lunch (30% of the RCV)</strong></td>
<td></td>
</tr>
<tr>
<td>Stewed rice &amp; Braised veal in vegetable sauce</td>
<td>10/Prot: 0.1 g; AC: 0 g; EAC: 0 g; P: 1 mg; K: 9 mg</td>
</tr>
<tr>
<td>Cucumber salad &amp; Cooked asparagus</td>
<td>10/Prot: 0.1 g; AC: 0 g; EAC: 0 g; P: 1.5 mg; K: 13.5 mg</td>
</tr>
<tr>
<td>Muffin</td>
<td>5/Prot: 2.4 g; AC: 1.2 g; EAC: 1 g; P: 95 mg; K: 70.5 mg</td>
</tr>
<tr>
<td><strong>Afternoon snack (10% of the RCV)</strong></td>
<td></td>
</tr>
<tr>
<td>Pancake</td>
<td>90/Prot: 2 g; AC: 2.1 g; EAC: 0.9 g; P: 47.5 mg; K: 76.2 mg</td>
</tr>
<tr>
<td>Honey &amp; Yeast</td>
<td>30/Prot: 0.1 g; AC: 0 g; EAC: 0 g; P: 1.5 mg; K: 13.5 mg</td>
</tr>
<tr>
<td>Yeast</td>
<td>5/Prot: 2.4 g; AC: 1.2 g; EAC: 1 g; P: 95 mg; K: 70.5 mg</td>
</tr>
<tr>
<td><strong>Dinner (20% of the RCV)</strong></td>
<td></td>
</tr>
<tr>
<td>Low-protein bread &amp; Sour cream</td>
<td>175/Prot: 2.9 g; AC: 2.9 g; EAC: 1 g; P: 56.1 mg; K: 106.4 mg</td>
</tr>
<tr>
<td>Cooked asparagus &amp; Cooked asparagus</td>
<td>50/Prot: 1.6 g; AC: 0.1 g; P: 42.5 mg; K: 66 mg</td>
</tr>
<tr>
<td>Green salad &amp; Cooked white of an egg</td>
<td>120/Prot: 2 g; AC: 0 g; EAC: 0 g; P: 44.4 mg; K: 163.2 mg</td>
</tr>
<tr>
<td><strong>Caloric value (MJ/kcal)</strong></td>
<td>10/2373</td>
</tr>
<tr>
<td>kcal/kg TT</td>
<td>34</td>
</tr>
<tr>
<td><strong>Nutritional value</strong></td>
<td></td>
</tr>
<tr>
<td>Protein (g/kg of body weight)</td>
<td>0.6 (0.55–0.6)</td>
</tr>
<tr>
<td>Amino acids</td>
<td>19.1</td>
</tr>
<tr>
<td>Essential amino acids (% of all the amino acids)</td>
<td>40</td>
</tr>
<tr>
<td>Essential amino acids (g/kg TT)</td>
<td>0.12</td>
</tr>
<tr>
<td>Branched-chain amino acids (%)</td>
<td>20.5</td>
</tr>
<tr>
<td>Isoleucine (g)</td>
<td>1 (0.6)</td>
</tr>
<tr>
<td>Leucine (g)</td>
<td>1.7 (0.9)</td>
</tr>
<tr>
<td>Valine (g)</td>
<td>1.2 (0.6)</td>
</tr>
<tr>
<td>Conditionally essential aminoacids (% of all the amino acids)</td>
<td>13</td>
</tr>
<tr>
<td><strong>Fats</strong></td>
<td></td>
</tr>
<tr>
<td>Percentage of energy (%)</td>
<td>29</td>
</tr>
<tr>
<td>Saturated fatty acids (%)</td>
<td>7.1 (&lt;10)</td>
</tr>
<tr>
<td>Monounsaturated fatty acids (%)</td>
<td>9 (&lt;10)</td>
</tr>
<tr>
<td>Polyunsaturated fatty acids (%)</td>
<td>6.5 (3–7)</td>
</tr>
<tr>
<td><strong>Carbohydrates</strong></td>
<td></td>
</tr>
<tr>
<td>percentage of energy (%)</td>
<td>63.7</td>
</tr>
<tr>
<td>Total dietary fiber (g)</td>
<td>30 (24 oz. &gt;30)</td>
</tr>
<tr>
<td><strong>Water-soluble vitamins</strong></td>
<td></td>
</tr>
<tr>
<td>Vitamin C (mg)</td>
<td>187 (110)</td>
</tr>
<tr>
<td>Vitamin B6 (mg)</td>
<td>1.1 (1.3)</td>
</tr>
<tr>
<td>Phospholipid (µg equivalent)</td>
<td>334 (431)</td>
</tr>
<tr>
<td>P (mg)</td>
<td>765 (&lt;1000)</td>
</tr>
<tr>
<td>K (mg)</td>
<td>1804 (&lt;2000)</td>
</tr>
<tr>
<td>Zn (mg)</td>
<td>4 (7.6)</td>
</tr>
<tr>
<td>Se (µg)</td>
<td>27 (30–70)</td>
</tr>
<tr>
<td>Na (mg)</td>
<td>2051 (1800–2500)</td>
</tr>
<tr>
<td>I from the iodized cooking salt (µg)</td>
<td>55 (220)</td>
</tr>
<tr>
<td>Water (ml)</td>
<td>1337 (&lt;2000)</td>
</tr>
</tbody>
</table>

Notes:
- Prot—protein; AC—amino acids; EAC—essential amino acids; P—phosphorus; K—potassium.
- Arginine, cysteine, glycine and tyrosine.
- Isoleucine, leucine, lysine, methionine, phenylalanine, threonine, tryptophan and valine.
- Essential amino acids: isoleucine, leucine, lysine, methionine, phenylalanine, threonine, tryptophan and valine.
meaning that the replacing procedure (LO in Fig. 1) starts replacing the elements from the last front and ends replacing the elements from the Pareto-optimal front. As a matter of fact, because the procedure is greedy, the replacement is stopped as soon as the first improvement is achieved. In this way, the cost of repair is minimized. Parameter settings: The parameters of the NSGA-II procedure, including the types of operators, are specified in Table 1. Detailed description of the operators is beyond the scope of the paper.

4.1.6. Time complexity

The basic operations and their worst-case complexities are as follows:

1. Non-dominated sorting of individuals is \(O(M(2N)^2)\) and of all infeasible individuals’ elements is \(O(M(2L_c)^2)\), where \(M\) denotes the number of objectives (1), \(N\) the population’s size, and \(L_c\) the length of a chromosome;
2. Crowding distance assignment is \(O(M(2N)\log(2N))\);
3. Sorting on the crowding comparison operator is \(O(2N\log(2N))\); and
4. Although it is known that specific variants of the simplex method require exponential time in the worst case (Megiddo, 1987), our empirical results have shown that in our case the LP complexity is \(O(\text{NCo}^2)\), where \(\text{NCo}\) denotes the number of constraints on the meal-planning sub-problem.

The overall complexity of EA for menu planning that solves \(L\) sub-problems including the global problem of \(n\)-day menu planning is \(O(LM(N^2))\), which is governed by the non-dominated sorting part of the algorithm.

5. Empirical results and discussion

The evolutionary method for menu planning has already been applied to the redesign of sample menus for children, workers and patients with special nutrition needs, which have recently been published in the Slovene Guidelines for Child Nutrition (Ribič Hlastan et al., 2008), the Slovene Guidelines for Workplace Nutrition (Pokorn et al., 2008), and the Slovene Recommendations for Clinical Nutrition and Nutrition for Elderly People in Care Homes (Rotovnik Kozjek et al., 2008), respectively.

The evolutionary method outperformed professionals in terms of time and quality. While it takes an experienced nutritionist or dietician from 30 min to 3 h to manually plan a daily menu for an individual or a group of individuals, a computer (1.7 GHz PentiumM, 512 MB RAM, Apache/PHP) needed from several minutes to a couple of hours to design a well-converged and well-distributed Pareto-optimal front of well-balanced optimal 21-day menus, depending on the complexity of the problem instance. The analysis showed a large percentage of infeasible solutions (an average 86%) in all sub-problems that we managed to reduce by LO and LP (an average to 34%).

We selected the most appropriate solutions from the Pareto-optimal fronts according to decisions made by a human after the search.

In Table 2, an example of a daily menu for patients with chronic kidney disease selected from the Pareto-optimal front developed by the evolutionary method is given. The menu was optimized upon a collection of daily menus for healthy adults, considering the following specific constraints (Rotovnik Kozjek et al., 2008):

- **Recommended value for protein**: 0.55–0.6 g/kg of body weight;
- **Recommended value for potassium (K)**: 1500–2000 mg.

The caloric and nutritional values were calculated using food composition data from the national FCDB for meat and meat products, the Souci–Fachmann–Kraut FCDB (Scherz and Senser, 2000) and the USDA FCDB, and a recipe method recommended by INFOODS that applies weight yield and retention factors published by Bognár (2002).

Acknowledgments

The author gratefully acknowledges the contributions of Profall Drážigost Pokorn, MD, and the support provided by the Slovenian Ministry of Health during the course of this work.

References


420