Portfolio Selection and Scheduling in the Power Generation Industry

Cleber Mira[1], Maria Angélica Souza[1], Arnaldo Moura[2], João Meidanis[1,2], Gabriel A. Costa Lima[4], Renato P. Bossolan[3] and Ítalo Freitas[3]

[1]Scylla Bioinformatics
[2]Institute of Computing, Unicamp
[3]AES-Tietê
[4]Mechanical Engineering Department, Unicamp

June 24, 2013

Abstract

The project portfolio selection (PPS) problem consists in choosing from a set of projects the ones that should be executed and deciding when they should start, depending on several restrictions involving project costs, limited resources, dependencies among projects, and aiming at different, and even conflicting, goals. We present a GRASP implementation for solving a particular version of the PPS problem derived from the electrical power generation industry. The problem includes distinct categories of resources and a dependency between projects. Solutions produced by the heuristic are around 10% better than those generated manually by experts.

keywords: Metaheuristics. Project portfolio. Greed algorithms.
Main area: OR in Administration & Production Management. OR in Energy. Metaheuristics

1 Introduction

Decision makers are usually confronted with the problem of choosing a subset from a large number of projects, that is, a project portfolio, according to a combination of criteria, and given a limited amount of resources. The construction of a project portfolio involves the selection and scheduling of projects over a period of time. This is known as the *project portfolio selection* (PPS) problem. There are several variant formulations for the PPS problem, depending on specific constraints, objective functions, decision variables, resource availability policies, and timing assumptions (Carazo et al. (2010)). However, since these variants usually share some characteristics, a single-objective PPS problem can be summarized as follows: find an instance of a project portfolio, that is, a selection of and a schedule of projects, that maximizes a given objective function over all viable portfolios.

The single objective function version of the PPS problem is a classical optimization problem (Baker (1974)), with applications in finances, see Campbell and Campbell (2001), research and development (Meade and Presley (2002); Mohanty, Agarwal, Choudhury, and Tiwari (2005)), software development (Lee and Kim (2000)), and project management (Archer and Ghasemzadeh (1999); Martinsuo and Lehtonen (2007)). There is an extensive literature involving several approaches to model PPS problem

variants, such as Fox, Baker, and Bryant (1984); Anandalingam and Olsson (1989). Even though it is not as studied as the single-objective version, the multi-objective version of the PPS problem has gained an increasing interest in recent years, e.g. Armananzas and Lozano (2005); Chiam, Tan, and Al Mamum (2008); Gutjahr, Katzensteiner, Reiter, Stummer, and Denk (2010). For a list of methods employed in the single- and multi-objective versions of the PPS problem, see Mira et al. (2012).

In this work, we approach a PPS problem occurring in the electrical power generation industry. For a company in this industry, the resolution of the PPS problem is essential for achieving strategical goals, complying with government regulations, and planning technical operations. Even small improvements in the scheduling of projects may have significant impact on the costs of large companies in this sector. We propose a GRASP meta-heuristic implementation for solving a particular PPS problem derived from the electrical power generation industry. The algorithm consists of a number of iterations of a two-phase procedure. The first phase uses a randomized greedy procedure to construct a viable portfolio. In the second phase, this portfolio is used as the initial solution for a local search procedure that continually improves the current solution until a locally optimal portfolio is found. The best solution over all iterations is then returned as the final result.

The paper is organized as follows. In Section 2, we describe the particular PPS problem studied here. In Section 3, we present details of the algorithm designed to solve the PPS problem. Section 4 describes a series of experiments performed to assess the quality and robustness of the algorithm, and analyses the results thus obtained. We summarize and point to further research directions in Section 5.

2 Problem description

We describe in this section a specific PPS problem derived from a real case occurring at the large electrical power generation company AES-Tietê, operating in the state of São Paulo, Brazil.

A power generation company is periodically confronted with the selection and scheduling of projects that contribute to achieve diverse, usually conflicting, objectives, such as reduction of operational costs, increase in investment return, and compliance with government regulations. Moreover, generating electricity is a complex process and, as such, is affected by several undesired and costly events. Often, these events result from improper operation, faulty equipment maintenance, or even from violations of government regulation and safety procedures. One technique that can be used to minimize the occurrence of undesired events consists in the execution of the following steps: (i) Elicit *undesired events* and identify the specific conditions that will trigger them; (ii) Assign to each undesired event a number that will amount to its *risk*; and (iii) Propose projects to control each undesired event. Risks are assessed according to a standard set of criteria, including probability of occurrence, severity of safety violations, environmental problems, harm on corporate image, and so on, in such a way that the risk of an undesired event should reflect the negative impact of its occurrence.

Each undesired event is then prevented by a *group of projects*. Only when all projects in a group are finished will the risk associated to the undesired event be considered completely under control. A project may belong to more than one group of projects and so it may contribute to control the risks associated with several undesired events. The amount of risk control associated to a project is the sum of all risk controlled by all groups the project belongs to. For instance, an undesired event can be the malfunctioning of a crucial machine due to lack of maintenance. So we need a group of projects than can guarantee proper maintenance for this machine: the acquisition of specific equipment and replaceable parts, personnel allocation for the maintenance activity, etc. The malfunctioning event will be sufficiently prevented when all projects in this group are over, and only then will the risk of this undesired event be totally controlled.

Of course, projects cannot be scheduled at will, due to constraints involving their costs and other available resources. Therefore, informally, the main goal in this particular PPS problem consists in *selecting and scheduling* projects that contribute to control as much as possible the risks associated to the undesired events, while keeping within specified limits on the available resources.

2.1 Problem Input

The input data provided by AES-Tietê is composed by a set of projects, each one not divisible into smaller activities. There is also a sequence of months, numbered from 1 to T, called the *planning horizon* (PH). In our case, $T = 60$, so the PH consists of five years. For each year of the PH a limit of *resources* is available for project execution. According to their main type of expenditure, projects are classified as either operational expenditures (OPEX) or capital expenditures (CAPEX).

Each project contains the following parameters:

(a) *Project identifier*: a unique, numeric identifier.
(b) The *duration* of the project, *i.e.* the number of months the project takes to be executed, which is lower than the total number of months within the PH.
(c) A sequence of *costs* describing the amount of resources a project needs at each month of its specified duration.
(d) In the input, each project is assigned an *initial starting month*. The whole set of initial starting months comprises a portfolio that was constructed manually. The values of the objective functions when computed over this manually constructed portfolio can serve as a yardstick against which we can measure the effectiveness of implementations.
(e) A *classification* designating the kind of each project: OPEX or CAPEX. A project may consume resources from only one of these two categories.
(f) A *project category* indicating whether a project is mandatory. If a project is mandatory, it must start at its designated initial starting month. A project can be classified as mandatory because of, for example, unavoidable engineering or weather constraints.

An undesired event is composed of the following parameters:

(a) An identifier: a unique, numeric identifier.
(b) A quantitative risk measure: indicating the risk associated to the undesired event.
(c) Group of projects: a list of project identifiers that defines a group of projects that guard against this undesired event. Some projects do not belong to any group and therefore do not contribute to risk control, *e.g.*, office supplies acquisition.

So, the input data for the PPS problem consists of a manually generated portfolio and a list of undesired events, each one of these being provided in a different spreadsheet. In the manually generated portfolio spreadsheet there is a column for each project parameter, one project per line. Similarly, each line of the undesired events spreadsheet specifies one of its parameters, the last column being a list of project identifiers that compose the group of projects guarding against the corresponding undesired event.

2.2 Decision variables and objective function

For ease of reference, let the projects be numbered 1 to I and the months within the PH be designated 1 to T. We use binary decision variables x_{it}, where $x_{it} = 1$ if and only if project i starts at month t, $1 \leq i \leq I$ and $1 \leq t \leq T$.

The objective function of the PPS problem studied here should reflect the best choice of scheduling projects in order to guard against undesired events. Hence, essentially what distinguishes one portfolio from another is the total amount of risk each one controls, and the time at which their corresponding undesired events are secured. Thus, the objective function should assign better values to portfolios that guard more undesired events, earlier in time, and with larger associated risks.

Let A be the set of undesired events in a PPS problem instance. Let I_a be the group of projects associated to undesired event $a \in A$. Denote by s_i the initial month a project is scheduled at, in other words, $s_i = t$ if and only if $x_{it} = 1$. Let $f(I_a)$ denote the first month when all projects in I_a are completed, that is, the month the associated undesired event is finally secured. Clearly, we have

$$f(I_a) = \max_{i \in I_a}(s_i + d_i - 1),$$

where d_i is the duration of project i.

An undesired event risk contributes to a portfolio overall controlled risk from the month it is secured. The *contribution* of securing an undesired event is given by the amount of risk associated to the undesired event times the number of months from when the undesired event is secured up to $2T$ months. We adopt $2T$ instead of T to handle the cases when the undesired event is only secured outside the PH. The objective function of a given portfolio is then:

$$\sum_{a \in A}(R_a(2T - f(I_a))), \tag{1}$$

where R_a is the amount of risk associated to undesired event a.

2.3 Constraints

We now describe the constraints associated to the PPS problem treated here.

Eventuality. Since project cannot start at two different months, we have:

$$\sum_{t=1}^{T} x_{it} \leq 1, \qquad \text{for } i = 1, \ldots, I. \tag{2}$$

Mandatory projects. Projects responsible for strategic decisions, regulatory demands, or small, recurring expenses have their scheduled time pre-defined and cannot be re-scheduled by the decision maker. In our model, all these projects are considered *mandatory*. A mandatory project i should start at its prescribed initial month p_i. In fact, the heuristic deals with mandatory projects by immediately scheduling them and subtracting the resources they consume from the available resources, before the optimization procedure starts. If M is the set of mandatory projects, we have:

$$x_{i,p_i} = 1, \qquad \text{for all } i \in M. \tag{3}$$

Limited resources. Each project demands a certain amount of resources, *i.e.*, it has a certain cost, for each month of its duration. The cost of a project i at month t of its activity is denoted by c_{it}. Projects are also classified into one of two kinds: OPEX or CAPEX. Denote by $C(t, q)$ the total amount of resources demanded by all active projects of kind q that are active at month t. Thus:

$$C(t, q) = \sum_{k=1}^{t} \sum_{i \in q} c_{i,(t-k+1)} x_{ik}, \qquad \text{for all } t \in T \text{ and } q \in \{\text{OPEX}, \text{CAPEX}\}. \tag{4}$$

For management reasons, resources are grouped by year. Let $W_{y,q}$ denote the available amount of resources for projects of kind q in year y, where $1 \leq y \leq T/12$. The sum of the costs for all projects at all months of a year must not surpass the available resources for that year, and per project kind. So, we have:

$$\sum_{j=12(y-1)+1}^{12y} C(j,q) \leq W_{y,q}, \qquad \text{for all } 1 \leq y \leq T/12,\ q \in \{\text{OPEX, CAPEX}\}. \tag{5}$$

Notice that there are no constraints that restrict project resources at month $T+1$ or later.

3 A GRASP Variant for the PPS Problem

We propose a variant of the GRASP meta-heuristic for solving the PPS problem treated here. GRASP is a multi-start meta-heuristic that generates good quality solutions for many combinatorial optimization problems (Resende (2009); Festa and Resende (2011)). The algorithm repeats two phases: (i) A construction phase in which a feasible solution is obtained; and (ii) a local search phase in which the algorithm finds a local optimal solution by iteratively checking whether there is a better solution in the neighborhood of the current solution. Several initial feasible solutions are constructed and improved upon until the best overall solution is then chosen.

Our algorithm uses a slightly modified version of the construction phase. Due to the specificities of the real PPS problem treated here, experiments with the adaptive character of the meta-heuristic showed that it could be simplified without sacrificing solution quality, while allowing for considerable gains in time performance (more details in Section 3.1). For this reason, we decided to replace the full adaptive greedy component of the heuristic by a simpler, greedy randomized, strategy. The pseudo-code for the GRASP routine is presented as Algorithm 1.

Algorithm 1 GRASP routine for the PPS problem

```
1. Initialize an empty portfolio as the current BEST_SOLUTION.
2. for R iterations do
3.       Construct a first solution P for the PPS problem. // Construction phase
4.       Let P = LOCAL_SEARCH(P) // Local search phase
5.       if P is better than BEST_SOLUTION then
6.             BEST_SOLUTION = P
7.       end if
8. end for
9. return BEST_SOLUTION.
```

3.1 Construction Phase

The construction phase procedure is based on the idea of sorting all possible pairs (i,t) of projects $i \in I$ and months $t \in [1,T]$, according to a suitable key parameter. The key parameter associated to a pair (i,t) should measure the contribution of scheduling project i at month t, based not only on i's controlled risk, but also on its total cost, and on its contribution to the control of undesired events.

The controlled risk of a project alone is not informative enough to decide whether it should be scheduled earlier than another project, without comparing also the projects' costs. A project with a large value of controlled risk may be less cost-effective in terms of controlled risk by unit of cost than another project (Mira et al. (2012)). The month at which a project starts to control risks is also relevant, since the earlier all projects in that group are scheduled, the earlier undesired events are secured and the larger their contribution to the objective function, since a undesired event risk is controlled only

when all projects in the associated group are terminated. So, the individual contribution of a project i scheduled at a month t could be measured by $(2T - t - d_i + 1)R_i$, where d_i is project i's duration and R_i is the sum of risks project i contributes to when controlling undesired events.

We now compare adaptive and non-adaptive procedures when used in this variant of the PPS problem.

An adaptive procedure. As a first attempt to estimate the cost-effectiveness of a project i being scheduled at certain month t, we adopted as a measure of its effectiveness the ratio between its contribution and the current *maximum resource usage* (MRU), *i.e.*, the maximum value of yearly available resources taken throughout the PH. Informally, the more a project consumes resources along its duration, the larger must be its controlled risk to compensate for not scheduling other projects due to lack of resources. When a project is scheduled and its costs are subtracted from the available resources, the maximum value of the remaining available resources may change, and this requires a reordering of the remaining candidate pairs (p, t), before the next iteration of the construction phase. This observation provides a mechanism for updating the candidate list after a pair (i, t) is chosen: for each remaining pair in the list, we recalculate its ratio between its contribution and the MRU, and sort the candidate list again, according to the new values. For the update and reordering of the candidate list, we also considered the effect towards the risk a project contributes when guarding against undesired events. The reasoning here is the following: as a project in a group that guards against an undesired event is scheduled, the remaining projects in the group should gain some priority over other projects, since the sooner they are scheduled the earlier the corresponding undesired event is finally controlled. Let $i \in I_a$ be a project that guards against undesired event a, and suppose that i was just scheduled to start at a certain month. Then, for any unscheduled project $j \in I_a$, $j \neq i$, the new measure of its controlled risk after i is scheduled is recalculated as $R'_j = R_j(1 + \delta\epsilon)$, where δ is the fraction of projects already scheduled in I_a, and ϵ indicates an increase percentage. Typical values tested for ϵ ranged from 5 to 20.

A non adaptive procedure. A simple non adaptive procedure for the construction phase consists in building a candidate list of pairs (i, t) and sorting it by the cost-effectiveness of scheduling the project i at the month t. To assess the cost-effectiveness of a particular project, we define the *benefit* of a project i, scheduled to start at month t, $i \in I$ and $1 \leq t \leq T$, as:

$$b(i,t) = \Big((2T - t - d_i + 1)R_i\Big) / \sum_{k=1}^{T} c_{i,k}, \tag{6}$$

The function $b(i, t)$ expresses the amount of risk gained by unit of cost, weighted by the number of months remaining after the project terminates.

The construction phase initially pre-processes the input data to schedule the mandatory projects, and then constructs a portfolio by iteratively choosing a pair (i, t) at random among the first k pairs from a list of candidate pairs. The candidate list is formed by all pairs (i, t) sorted by their $b(i, t)$ value and such that the inclusion of the pair in the portfolio keeps the solution feasible. At each iteration in the first phase, as a pair (i, t) is chosen, the available resources at each year are decreased reflecting the resource usage of project i, and the cycle repeats.

We further restricted the candidate pairs (i, t) by taking t a multiple of 3. This was motivated by expert engineers, who observed that most large projects at the generation and distribution plants were typically best scheduled in cycles of three months. This also helped the heuristic to produce better than manual solutions, while still close to what engineers were accustomed to deal with, thereby much facilitating the use of the portfolio management prototype by them.

Table 1: Construction phase solutions using adaptive and non adaptive version of the heuristics.

DF	Non adaptive	List update w/o MRU	List update with MRU
0%	2649117.54	2437539.86	2269148.41
5%	2597818.88	2440933.15	2232109.75

We performed some experiments to compare adaptive *versus* non adaptive procedures for the construction phase. We set the window size to $k = 1$ to avoid interference from randomization on the experiments, and let $\epsilon = 15\%$, the best value found after empirical tunning. The experimental results are summarized in Table 1. The values in the table show the total amount of controlled risk, one input instance per line. Column two lists values using the non adaptive procedure, while columns three and four list values using the adaptive procedure, without and with the MRU strategy, respectively. Column one shows the disturbance factor. As we further comment in Section 4, in order to test the robustness of the algorithm we also created new input instances by adding small random perturbations to the real input instance. It can be seen from Table 1 that the non adaptive algorithm outperformed any other arrangement involving adaptive list updates. Moreover, this was the case for both the original and perturbed input instances. Therefore, we adopted a non adaptive algorithm in the construction phase. We summarize the non adaptive procedure used in the construction phase in Algorithm 2.

Algorithm 2 Construction phase for the PPS problem

1. Initialize P as an empty portfolio.
2. **for all** mandatory projects **do**
3. Schedule the mandatory project at their initial scheduled month.
4. Subtract mandatory project's resources from the available resources.
5. **end for**
6. Calculate the risk R_i of each project i.
7. Construct the SortedPairs (SP) list of remaining project $\times$ month pairs.
8. Sort SP by benefit value.
9. **while** SP is not empty **do**
10. Get a pair (i, t) randomly chosen from the first k pairs at the top of the SP list.
11. Remove pair (i, t) from SP.
12. **if** (i is not already scheduled) **and** (adding (i, t) keeps the solution feasible) **then**
13. Let $P = P \cup \{(i, t)\}$.
14. **else**
15. Discard pair (i, t).
16. **end if**
17. **end while**
18. The remaining projects are scheduled at month $T + 1$.
19. **return** Portfolio P.

The construction algorithm guarantees that any project not scheduled at the main loop of the construction phase will be scheduled at month $T + 1$. These projects are scheduled at $T + 1$ because at the portfolio management prototype that was implemented (Mira et al. (2012)), decision makers could alter the solution found by the software by modifying any project's start time and, so, it was necessary to keep all projects visible to the user.

The running time of the construction phase is thus of order $\Theta(n \log n)$, where $n = G \times \frac{T}{3}$ and with G the number of non mandatory projects and T the number of months in the planning horizon. Recall that projects are considered for scheduling in a cycle of 3 to 3 months.

3.2 Local search

The GRASP second phase consists of executing a local search procedure a certain number of times. The local search starts at the solution obtained in the construction phase and examines its neighborhood, checking whether there is a better solution in terms of the objective function. It iteratively repeats the search in the neighborhood of each new better solution for a predefined number of times. The local search procedure is shown as Algorithm 3.

Algorithm 3 Local search procedure

1. Let P be the initial solution.
2. **for** L iterations **do**
3. Get the list of neighbors N of the current solution P.
4. Search through N, using a First or Best Improvement strategy.
5. **if** an improved solution Q was found **then**
6. Let $P = Q$
7. **else**
8. **break**
9. **end if**
10. **end for**
11. **return** Portfolio P.

A fundamental aspect of a local search procedure concerns how the neighborhood of a solution is defined. We define the neighborhood of a portfolio as the set of portfolios where each neighbor is obtained by re-scheduling one of its projects. If a project was originally scheduled at month t, a neighbor is obtained by re-scheduling it at months from $t - \Delta$ to $t + \Delta$, avoiding the current t, and where Δ is a fixed parameter. If re-scheduling leads to infeasibility, or to a starting time beyond T, that neighbor is discarded and we proceed to test the next one. We note that only projects whose start times are in the interval $[1, T]$ are considered for re-scheduling. Therefore, a portfolio can have up to $2 \times H \times \Delta$ neighbors, where H is the current number of projects starting from months 1 to T in the current solution. Another relevant aspect of the local search is how the neighborhood is searched. We implemented two strategies for searching a neighborhood. In the *First improvement* strategy, the search is performed until the first neighbor is found that yields a better solution than the current one, while in the *Best improvement* strategy, the search goes through all the new solutions in the neighborhood and picks the best one. Clearly, the choice between the two strategies has an impact on the performance of the local search algorithm. Algorithm 4 show the procedure to construct the neighborhood of a portfolio solution. Since each local search comprises at most L iterations, the running time of the local search phase is of order $O(L \times \Delta \times H^2 \times T)$.

Algorithm 4 Returns the neighborhood of a solution

1. Let P be the initial solution.
2. Initialize an empty list of solutions N.
3. **for all** pairs (i, t) in P **do**
4. **for all** months m in the interval $\left[t - \Delta, t + \Delta\right] - \{t\}$ **do**
5. $Q = \left(P - \{(i, t)\}\right) \cup \{(i, m)\}$
6. **if** Q is feasible **then**
7. Let $N = N \cup \{Q\}$
8. **end if**
9. **end for**
10. **end for**
11. **return** List of neighbors N

4 Experiments

We performed experiments to evaluate the performance and robustness of the algorithm. We chose the heuristic running-time and the objective function value as good indicators of solution quality.

In order to present some of the real instance data while complying with AES-Tietê confidentiality policies, we performed a simple data obfuscation scheme on the costs and available resources present in the real data. The following is a summary of the real instance data, after the obfuscation procedure was applied:

- A measure of the cumulative controlled risk: 2,418,204.89.
- Number of projects: 1411.
- Planning horizon: years 2013, 2014, 2015, 2016, and 2017.
- Average cost of each project by year (in thousands of US$): 1082, 1073, 1323, 896, and 633.
- Total sum of costs for all years (in thousands of US$): 7.88 million.

The amount of available yearly CAPEX resources were obtained by summing up the CAPEX costs of all projects in the initial manual solution that were active at each year. Yearly OPEX available resources were obtained in a similar fashion.

4.1 Input data

In order to assess the indicator values and test the robustness of the heuristic, we compared the manual portfolio solution for a real-world instance against the solution generated by the algorithm for that same instance. The manual portfolio solution was built by expert managers and engineers at AES-Tietê. To assess the robustness of the procedure, we used the real-world instance provided by AES-Tietê as a seed for generating similar real world-like instances, by randomly disturbing the seed instance data, in a controlled way and by small margins.

The disturbing procedure applied the following modifications to the real instance data, where d was a fixed disturbance factor:

Project cost: modified by $x\%$, with x randomly chosen between $-d$ and $+d$.

Project risk: modified by $x\%$, with x randomly chosen between $-d$ and $+d$.

Other parameters: not modified in the disturbed instances.

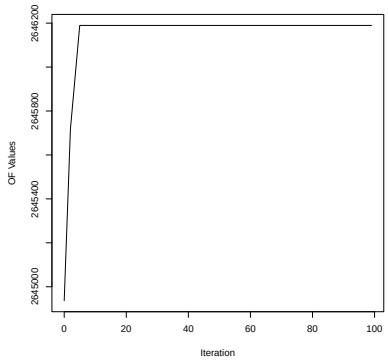
Using these modifications, we generated 5 instances for a 5% disturbance factor. We decided to adopt this disturbance factor because, according to engineers, modifications in the overall cost beyond 5% in a year are unlikely in a traditional industry such as in the electrical power generation industry, whose assets do not change by a large amount yearly.

4.2 Heuristic parameters

Before running the robustness experiment, we performed some calibration experiments in order to identify adequate values for some of the GRASP parameters.

During the construction phase, the next project and month pair, (i, t), to be inserted in the current partial solution was randomly chosen among the first k best candidates in the sorted candidate list. We set k to 5, because this relatively small value was a good compromise between enforcing the randomized aspect of the heuristic while not loosing the greedy advantage of having the candidate list sorted.

We experimented with how many local search iterations would be enough for finding a local optimal, in a typical run. For that, we executed the algorithm with the number of local search iterations set to 100, using the real instance as input, and identified that 5 iterations were typically enough to find the best local optimal solution. Figure 1 shows the increase in objective function values as the local search proceeded. With this observation, we chose 20 for the number of repetitions in the local search.



Iteration	OF value
1	2644936
2	2645327
3	2645875
4	2646032
5	2646189

Figure 1: Objective function value at each local search iteration, using *best improvement*.

4.3 Experimental results

The optimization procedure was executed 10 times for each input instance, and the best solution was retained for each instance. Each input instance was solved using two distinct local search techniques: first improvement and best improvement. The experiments were performed in a Intel(R) Core(TM) i5-2400 3.10GHz CPU, with 4Gb of RAM, in Ubuntu Gnu/Linux (kernel 3.2.0-40) and in an AMD Phenom(tm) IIX6, with 8Gb of RAM, in Ubuntu Gnu/Linux (kernel 3.2.0-27). The algorithm was implemented using the Python 2.7.3/Django 1.5 environment.

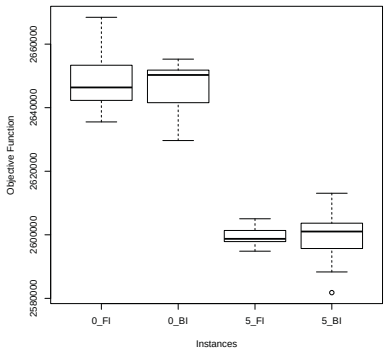
A summary of the results appears in Table 2. Each line reports the best and the average objective function values for 10 runs using the same input instance. The first column shows the Disturbance factor (DF) and an instance identification. A DF of 0% refers to the real instance. The second and third columns are, respectively, the average and the best optimized objective function (OOF) values, using the *first improvement* local search strategy. The following two columns are the average and maximum OOF values using the *best improvement* local search strategy. The last column contains the objective function value calculated for the input instance (IOF). The average total running time for executing 10 runs for each instance are around 60 minutes using first improvement, and about 100 minutes using best improvement. As can be seen from the table, the heuristic consistently found solutions whose objective function values were around 10% better when compared to corresponding values for the manual solution. This behavior was repeated when the algorithm took as input any of the five disturbed instances, attesting to its robustness when relatively small changes were applied to the real input instance. Also note, with respect to the *first improvement* and *best improvement*, that the algorithm did not seem to perform better when using one strategy than the other.

Figure 2 shows some best optimized (OOF) values statistics for two typical instances. The data is summarized in four box-plots: the first two present statistics for the OOF values obtained from the real instance using the first improvement (0_FI) and best improvement (0_BI) local search strategies, respectively. The second two box plots present the OOF statistics for a 5% perturbed instance, again

Table 2: Average and best solution values for two local search strategies

	First Improvement		Best Improvement		Manual
DF - instance	avg OOF	max OOF	avg OOF	max OOF	IOF
0%	2648369.43	2668440.21	2646078.77	2655274.94	2418204.89
5% - a	2599563.23	2605041.79	2599059.37	2613078.97	2377205.50
5% - b	2599985.21	2604749.89	2598710.27	2610761.26	2377205.50
5% - c	2605699.84	2620827.47	2600409.27	2604835.26	2377205.50
5% - d	2600259.51	2604927.08	2598337.45	2604320.26	2377205.50
5% - e	2599815.34	2604905.00	2599201.13	2617631.77	2377205.50

adopting the first improvement (5_FI) and best improvement (5_BI) local search strategies, respectively.



Instance	Min	1st Q	Median	Mean	3rd Q	Max
0_FI	2635521	2643029	2646370	2648369	2652858	2668440
0_BI	2629657	2641650	2650292	2646079	2651552	2655275
5_FI	2594845	2597880	2598727	2599563	2601020	2605042
5_BI	2581819	2596498	2601035	2599059	2603175	2613079

Figure 2: Best solution value statistics for the real instance and a perturbed instance.

5 Conclusion

We modeled a PPS problem derived from a large electrical power generation and distribution company, namely, AES-Tietê. The problem involves an intricate relationship among projects that guard against undesired events. As such undesired events are secured, the risk associated to them are controlled and their odds of occurring mitigated.

For solving this specific PPS problem, we proposed a variant of the GRASP meta-heuristic. We implemented the construction phase by a simpler greedy randomized procedure that picks the best project and start time pairs according to a benefit function, computed from the given data. For the local search phase, neighborhoods where obtained by displacing projects original start times by a small amount. Both first improvement and best improvement strategies were implemented in the local search phase. We performed experiments using real-world input instances in order to find appropriate parameters for the heuristics and to assess the algorithm robustness. The heuristics showed a significant improvement over the manual solutions, both with the real data and with the disturbed data as input.

Future work will involve the inclusion of new constraints concerning the impact of projects on the operation of electrical power plants, different categories of available resources, such as human resources, the concurrent use of multiple objective functions, and other, more sophisticated adaptive procedures to be used in the algorithm construction phase.

Acknowledgment

The authors would like to thank AES-Tietê for their technical support and real data sets, and thank the National Agency for Electrical Energy (ANEEL) for their financial support (grant PD 0064-1021/2010).

References

Anandalingam, G., & Olsson, C. (1989). A multi-stage multi-attribute decision model for project selection. *European journal of operational research*, *43*(3), 271–283.

Archer, N., & Ghasemzadeh, F. (1999). An integrated framework for project portfolio selection. *International Journal of Project Management*, *17*(4), 207–216.

Armananzas, R., & Lozano, J. (2005). A multiobjective approach to the portfolio optimization problem. In *Evolutionary computation, 2005. the 2005 ieee congress on* (Vol. 2, pp. 1388–1395).

Baker, N. (1974). R&D project selection models: an assessment. *IEEE Transactions on Engineering Management*, *21*(4), 165–171.

Campbell, J., & Campbell, R. (2001). *Analyzing and managing risky investments*. CPI.

Carazo, A., Gómez, T., Molina, J., Hernández-Díaz, A., Guerrero, F., & Caballero, R. (2010). Solving a comprehensive model for multiobjective project portfolio selection. *Computers & operations research*, *37*(4), 630–639.

Chiam, S., Tan, K., & Al Mamum, A. (2008). Evolutionary multi-objective portfolio optimization in practical context. *International Journal of Automation and Computing*, *5*(1), 67–80.

Festa, P., & Resende, M. (2011). GRASP: basic components and enhancements. *Telecommunication Systems*, *46*(3), 253–271.

Fox, G., Baker, N., & Bryant, J. (1984). Economic models for R and D project selection in the presence of project interactions. *Management science*, 890–902.

Gutjahr, W., Katzensteiner, S., Reiter, P., Stummer, C., & Denk, M. (2010). Multi-objective decision analysis for competence-oriented project portfolio selection. *European Journal of Operational Research*, *205*(3), 670–679.

Lee, J., & Kim, S. (2000). Using analytic network process and goal programming for interdependent information system project selection. *Computers & Operations Research*, *27*(4), 367–382.

Martinsuo, M., & Lehtonen, P. (2007). Role of single-project management in achieving portfolio management efficiency. *International Journal of Project Management*, *25*(1), 56–65.

Meade, L., & Presley, A. (2002). R&D project selection using the analytic network process. *Engineering Management, IEEE Transactions on*, *49*(1), 59–66.

Mira, C., Feijão, P., Souza, M., Moura, A., Meidanis, J., Lima, G., ... Freitas, I. (2012). A GRASP-based heuristic for the Project Portfolio Selection Problem. In *The 15th IEEE International Conference on Computational Science and Engineering (IEEE CSE2012)* (Vol. 1, pp. 36–41).

Mohanty, R., Agarwal, R., Choudhury, A., & Tiwari, M. (2005). A fuzzy ANP-based approach to R&D project selection: a case study. *International Journal of Production Research*, *43*(24), 5199–5216.

Resende, M. G. C. (2009). Greedy Randomized Adaptive Search Procedures. In *Encyclopedia of optimization* (p. 1460-1469). Springer.