# S.A.R.A.: Uma Ferramenta Baseada em Algoritmos Genéticos para Alocação de Salas de Aula do IFBA

Ícaro Jerry Salles Santana
Instituto Federal da Bahia
Salvador – Bahia – Brasil
icarojerry@gmail.com

Flávia Maristela Santos Nascimento
Instituto Federal da Bahia
Salvador – Bahia – Brasil
flaviamsn@ifba.edu.br

***Abstract*—The resources allocation problem has already been dealt with by many approaches in literature. Manually, such approach can be exhausting, time and resoureces demanding and produce unefficient solutions. This work presents an automation tool for allocationg classes on classrooms, considering some institutional restrictions. This solution uses Genetic Algorithms since it is an efficient and flexible meta-heuristic. The study case assumed System Development and Analysis Technology course at Federal Institute of Bahia - *Campus* Salvador.**

***Keywords*—*Combinatorial Optimization; Class Assignment Problem; Genetic Algorithms***

***Resumo*—O problema de alocação de recursos já foi tratado com diversas abordagens na literatura. A abordagem manual pode ser exaustiva, demandar tempo, recursos e as soluções geradas podem não ser eficientes. Este trabalho apresenta uma ferramenta para automatizar o processo de alocação de aulas em salas, considerando algumas restrições institucionais. A solução usa Algoritmos Genéticos por ser uma meta-heurística eficiente e flexível. O estudo de caso foi feito considerando o curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal da Bahia - *Campus* Salvador.**

***Palavras-chave*—*Otimização Combinatória; Problema de Alocação de Salas; Algoritmos Genéticos***

## I. Introdução

A alocação de recursos é um problema de Otimização Combinatória e, no geral, possui diversas aplicações no mundo real. Nas Instituições de Ensino Superior (IES) há diversas atividades de alocação de recursos, que consequentemente podem ser resolvidas como um problema de otimização. A designação de grade horária, turmas em salas e professores em disciplinas são exemplos desse tipo de atividade.

Parte das IES ainda tratam os seus problemas de alocação de recurso de forma manual [1], [2], [3], [4]. Dependendo da quantidade de recursos disponíveis, esta abordagem pode ter grande complexidade temporal, o que significa demandar um tempo considerável para ser concluída. Além disso, alterações posteriores podem gerar grande retrabalho e/ou tornar as soluções ineficientes. Isso ocorre, principalmente, porque a solução final deve respeitar um conjunto de restrições de forma a atender as necessidades das partes, neste caso alunos, professores e a própria instituição.

Neste trabalho abordamos a questão do problema de alocação de salas, também denominado como *Class Assignment Problem* [5]. Trata-se de um problema da classe NP-Difícil, e por isso a utilização exclusiva de algoritmos determinísticos torna-se inviável. Vários fatores críticos envolvem a resolução dos problemas desta natureza, tais como, o elevado número de combinações e o gerenciamento das restrições, o que implica diretamente no crescimento do tempo utilizado para obter uma solução factível [6].

Em casos como este, é mais viável verificar se uma solução está correta e determinar o quão boa ela é, do que tentar obter tal solução analiticamente em tempo polinomial [4]. Nesse contexto as meta-heurísticas ganham importância uma vez que são comumente utilizadas em problemas de otimização combinatória, devido a sua capacidade de escapar de soluções sub-ótimas (ótimos locais) [7].

Diversas meta-heurísticas estão disponíveis na literatura e se mostram soluções estabelecidas para melhorar este cenário [8]. Dentre elas, destacam-se a *Simulated Annealing*, Busca Tabu e Algoritmos Genéticos [6], [9]. Para lidar com o problema de alocação de salas no IFBA usamos o método dos Algoritmos Genéticos por ter relatos da eficiência de sua aplicabilidade e por apresentar uma arquitetura simples, capaz de se adaptar bem na resolução de diversos tipos de problemas [1], [4], [7], [10], [11], [12].

Existem algumas soluções para este problema que foram baseadas em Algoritmos Genéticos. Entretanto, os critérios de alocação podem mudar de acordo com o conjunto único das diretrizes de cada IES. Por esse motivo os sistemas disponíveis para mapeamento de turmas em salas de aula se tornam inviáveis quando aplicados em contextos genéricos. Sendo assim, as ferramentas foram desenvolvidas de forma a se adequar às necessidades específicas da instituição em que será aplicada [10].

Este trabalho apresenta a S.A.R.A. – *Smart and Agile Resource Allocator*, uma ferramenta que visa otimizar o processo de alocação de turmas em salas de aula do Instituto Federal da Bahia (IFBA). Uma vez que, até dado momento, a instituição não faz uso de ferramentas para automatizar esta tarefa, realizando toda a alocação de forma manual.

A ferramenta proposta busca gerar soluções factíveis para o Problema de Alocação de Salas, a um baixo custo computacional, satisfazendo o máximo dos interesses do IFBA. Para isso, foi utilizado o cenário real do curso de Análise e Desenvolvimento de Sistemas da própria instituição. Uma série de testes foram realizadas visando analisar o desempenho da ferramenta e a qualidade das soluções geradas.

O restante deste documento está organizado como se segue. Na seção II são apresentados os fundamentos teóricos. A seção III descreve os trabalhos correlatos. A solução proposta, bem como detalhes relacionados ao problema em questão são apresentados na seção IV. Os testes e a validação são apresentados na seção V. Por fim, a seção VI conclui o trabalho realizado, apresentando as suas limitações e os trabalhos futuros.

## II. Referencial Bibliográfico

### A. Otimização Combinatória

Otimização pode ser definida como a busca da melhor opção disponível em uma ampla variedade de possíveis escolhas, visando maximizar (ou minimizar) a função objetivo. A dinâmica deste processo consiste em tentar encontrar a solução ótima repetidas vezes, utilizando informações obtidas durante o processo, a fim realizar uma melhora iterativa nas soluções.

Por exemplo, ao tentar sincronizar uma estação de rádio se guiando pela qualidade do som, está sendo feita uma otimização. Através do ajuste manual, várias soluções são testadas, até que o resultado esteja ótimo, neste caso o som com nitidez.

Problemas de otimização podem ser divididos entre três categorias: Otimização Contínua, Otimização Combinatória ou Discreta, e Otimização Mista [13]. O problema abordado neste trabalho é um problema de otimização combinatória, o que significa dizer que as variáveis de decisão assumem valores discretos [13]. A Otimização Combinatória é comumente utilizada na resolução de aplicações reais complexas, onde geralmente a solução pertence à um conjunto discreto, resultante de todas as combinações possíveis [14].

### B. Problema de Alocação de Salas

Um problema de alocação de recursos pode ser mapeado como um problema de otimização combinatória. Em situações em que dada uma quantidade de recursos disponíveis, pretende-se encontrar a melhor forma de distribuir as diversas atividades independentes, com base na qualidade da função objetivo.

No contexto das IES, um problema de alocação de recursos possui várias aplicações práticas, dentre elas podemos destacar o escalonamento dos horários de aula (*Course Timetabling*), programação da grade de exames (*Examination Timetabling*) e a designação das turmas em salas de aula (*Classroom Assignment Problem*) [5].

O problema de alocação de salas de aula é definido por [5] como uma variante do *Course Timetabling*. Tal problema consiste em alocar aulas, com horários de início e término previamente programados, a um número limitado de salas, respeitando um conjunto de restrições e satisfazendo o máximo de preferências [5], [7]. A Figura 1 ilustra duas turmas distintas com duas aulas cada, sendo cada aula dessas turmas, alocadas à um elemento denominado *slot*, resultante da combinação entre um dia e um horário, e uma sala.

No contexto da designação de salas de aula, uma boa solução é aquela em que há um bom aproveitamento do espaço físico, uma melhor eficiência na distribuição dos recursos e concordância no funcionamento dos horários de aula.

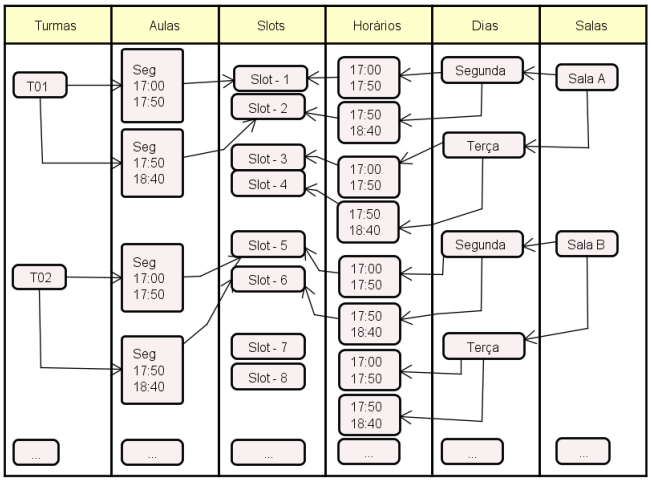


Figura 1. Combinação entre duas aulas e duas salas.

Além disso, uma boa solução deve atender adequadamente as restrições, resultando em aulas com suporte necessário de requisitos, tais como equipamentos e acessibilidade, melhorando a satisfação dos discentes e docentes.

O problema de alocação de sala é classificado como NP-Difícil e é caraterizado por possuir, geralmente, um amplo conjunto de elementos e restrições [7]. Por apresentar tais características, pode ser inviável encontrar soluções válidas através de métodos convencionais de programação, em um tempo computacional aceitável [5], [7].

O IFBA, bem como grande parte das IES [1], [2], [3], [4], trata de forma manual os problemas de alocação de recursos, um método exaustivo de busca. O uso deste método gasta várias horas, dias ou até semanas, além de várias pessoas dedicadas em buscar uma solução que seja viável.

Utilizar técnicas exaustivas pode ser uma tarefa custosa, ou até mesmo inviável, mesmo para problemas de pequena instância, pois o número de combinações geradas pode ser elevado [6]. É possível observar na Figura 1, a combinação entre duas salas, dois dias e dois horários, resultaram em oito *slot*.

Em ambientes dinâmicos, o conjunto de especificações e propriedades do cenário se modificam ao longo do tempo, podendo inviabilizar uma solução em um curto período. Caso seja resolvido através de algoritmos de força bruta o custo computacional é da ordem de $O(n!)$ [6]. Além disso, as particularidades envolvidas em cada IES dificultam mais encontrar soluções factíveis.

### C. Heurísticas e Meta-heurísticas

Heurísticas são estratégias que guiam o processo de busca por boas soluções, a um custo computacional aceitável. Em problemas de grandes instâncias ou quando o tempo para obter uma solução é um fator crítico, algoritmos heurísticos se mostram como uma boa alternativa, uma vez que apresentam soluções aproximadas e que podem atender completamente as restrições do problema. Algoritmos Gulosos, *Branch and Bound* e *Backtracking* são exemplos de métodos heurísticos [9].

Entretanto, tais técnicas não garantem encontrar a melhor solução de todo o problema (ótima global). Além disso, pode não ser possível calcular o quão próxima a solução atual (encontrada pela heurística) está da solução ótima (GAP) [15].

Métodos heurísticos podem produzir bons resultados, porém nos cenários reais das IES, o tamanho e complexidade atuais têm provocado uma tendência para algoritmos meta-heurísticos [1]. As meta-heurísticas [9] são técnicas que independem do problema, tornando-se assim mais indicadas a serem utilizadas como mecanismos genéricos de otimização.

Essas técnicas são comumente utilizadas para encontrar soluções em problemas com pouca informação, grande espaço de busca (região onde se encontram as soluções viáveis) e não se sabe qual deve ser a solução ótima. Entretanto, é possível avaliar uma dada solução, testá-la e definir o quão boa ela é. Muitos problemas de Otimização Combinatórias possuem essas características, tornando a meta-heurística um método relevante para resolver problemas da área [6].

As meta-heurísticas utilizam de estratégias probabilísticas e uso de informações de resultados anteriores, para se guiar e realizar a pesquisa no espaço de busca. Sendo assim, essas técnicas evitam convergir antecipadamente em soluções próximas à ótima (ótimos locais). São exemplos de algoritmos meta-heurísticos o *Simulated Annealing*, Busca Tabu e os Algoritmos Genéticos [9], [12].

### *D. Algoritmos Genéticos*

Os Algoritmos Genéticos (AGs) são métodos de busca inspirados na dinâmica evolucionária das espécies, segundo as teorias de Charles Darwin e Gregor Mendel [12]. Os AGs se diferem dos métodos tradicionais determinísticos de otimização, pois utilizam meios probabilísticos para convergir a um resultado. Cada possível solução é analisada de forma a determinar o quão boa ela é [4], [16].

Os AGs operam sempre com uma *população*, buscando através de sucessivas *gerações*, resultados cada vez melhores para um determinado problema. Uma população é formada por um conjunto de *indivíduos*, que cada *indivíduo* é uma representação abstrata de uma solução. A cada iteração do *ciclo de evolução* é criada uma nova *geração de indivíduos*, devendo atender melhor aos objetivos da otimização que a anterior [17].

Os elementos básicos que compõem parte da estrutura dos AGs são representados na Figura 2 [12], [17]. Cada indivíduo é composto por uma estrutura de informações (representação de um indivíduo), chamadas de *cromossomo*, que por sua vez é composto por um ou mais *genes*. Cada *gene* possui um valor, denominado de *alelo*, no qual define sua variável de decisão.

A Figura 3 ilustra o funcionamento básico de um Algoritmo Genético [12], [17]. Para cada solução é atribuída um valor que define o quanto apto é o indivíduo para um determinado problema (***fitness***). Este valor se baseia no conjunto de regras e especificações definidas. O conjunto de indivíduos que possuem melhores *aptidões* são selecionados para serem os *genitores* de uma *nova geração*, chamada de *descendentes*.

Ao ser iniciado, a primeira etapa do AG consiste em "*gerar uma população inicial*". Diversas estratégias podem

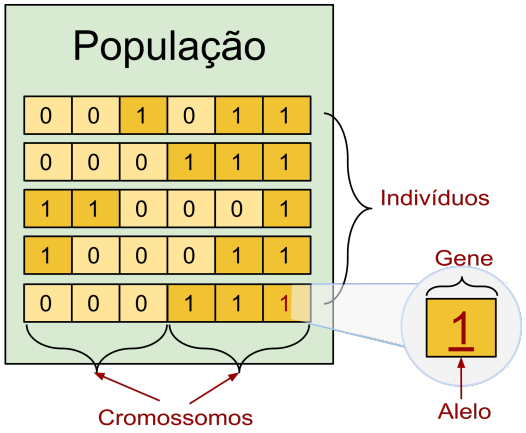


Figura 2. Elementos que compõem um Algoritmo Genético.

ser utilizadas para gerar a população inicial, sendo a uma das mais utilizadas, a forma aleatória, devido a simplicidade da sua implementação e por garantir maior diversidade dos indivíduos gerados [17].

Vale ressaltar que uma *população inicial* bem formulada pode melhorar consideravelmente o desempenho do AG [17]. Outro aspecto a ser levado em consideração é o tamanho da *população*. Uma população grande irá utilizar mais ciclos de processamento e memória, tornando o AG lento. Em contrapartida, a busca se torna mais completa, verificando mais áreas do espaço de busca. Já uma população pequena torna o AG mais rápido e com baixo custo computacional, porém, a busca será menos completa dificultando encontrar boas soluções.

Com a população inicial criada, é então iniciado o "*ciclo de evolução*". Este ciclo tem como objetivo fazer com que a população inicial evolua para uma geração seguinte através dos seus *operadores* principais: "*seleção*", "*cruzamento*" e "*mutação*".

No início do ciclo de evolução, durante a fase da "*avaliação da população*", é atribuído a cada indivíduo da *população atual* o seu respectivo *fitness*. O *fitness* representa a qualidade da possível solução em relação às características do problema. Este valor é especificado pela função de avaliação (ou função objetivo), cuja finalidade é definir o critério que avalia cada hipótese de acordo com o objetivo a ser atingido.

Na etapa da "*seleção*" os melhores indivíduos (aqueles com maior aptidão) terão mais chance de se reproduzir e passar suas características genéticas para as gerações seguintes. Entretanto, os indivíduos com menor aptidão podem conter características genéticas favoráveis a geração da melhor solução para o problema, considerando que tais características não estejam nos cromossomos da população atual.

A quantidade de indivíduos a serem selecionados é baseada em um parâmetro de configuração do AG. Uma taxa de seleção muito elevada pode fazer o algoritmo convergir precocemente, enquanto uma taxa muito baixa torna a evolução demorada. O operador de seleção pode ser implementado de diversas formas, tais como, *seleção proporcional*, *seleção por torneio* e *seleção por posicionamento* [17].

Após realizada a seleção dos indivíduos, uma nova geração

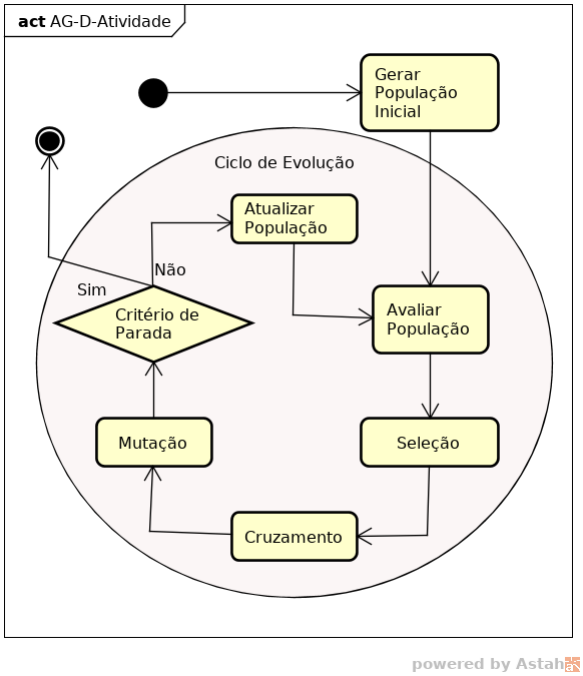


Figura 3. Funcionamento básico de um Algoritmo Genético.

é criada através da etapa de "*cruzamento*". Este operador realiza a combinação de partes do código genético entre dois indivíduos (genitores), gerando dois novos (descendentes). A forma como esses indivíduos são combinados variam de acordo com a técnica escolhida, sendo algumas delas, *cruzamento de um ponto*, *cruzamento de dois pontos* e *cruzamento uniforme* [17].

Ao decorrer das iterações, é possível que o AG venha a convergir para uma população uniforme, representando apenas uma pequena parte do espaço de busca. Sendo assim, a um custo computacional razoável, é desejável ter uma busca sobre um espaço maior, a fim de aumentar a probabilidade em obter uma solução mais qualificada.

Para evitar que isso ocorra, é executada a "*mutação*", etapa crucial do processo evolutivo, pois é responsável pela presença dos novos tipos de propriedades genéticas na população, garantindo a manutenção da diversidade da população. Os indivíduos gerados na etapa anterior sofrem uma alteração aleatória em seu código genético, baseado em uma probabilidade configurada previamente.

Assim como ocorre na natureza, este evento deve acometer uma pequena parcelas dos indivíduos da população. De outra forma, os indivíduos mutados pouco se assemelharam com seus genitores e o algoritmo irá se comportar de forma aleatória, podendo não convergir para soluções ótimas.

Feito isso, é verificado o "*critério de parada*", geralmente definidos pelo número máximo de gerações, qualidade mínima de uma solução, ou o limite do tempo de execução. Caso nenhuma condição de parada seja atingida, o AG se dirige para etapa de "*atualização da população*" e em seguida continua o *loop* passando a nova geração para etapa de seleção. Caso contrário, finaliza sua execução, apresentando a melhor solução encontrada durante todo o processo.

## III. Trabalhos Correlatos

Estão disponíveis na literatura diversos trabalhos que visam resolver os problemas de alocação de recursos existentes em uma IES [1], [2], [4], [10], [11], [18], [19], [20], [21]. Entretanto, estes modelos não podem ser aplicados no IFBA por conta do conjunto de restrições e especificidades de cada IES. A Tabela I exibe os principais aspectos destas soluções e a proposta deste trabalho.

Em [2] é proposto um sistema Web, automatizado, chamado Grade Horária, para problema de programação de horário do curso de Bacharelado em Sistemas de Informação (BSI) da Universidade Federal do Estado do Rio de Janeiro (UNIRIO). O trabalho buscou satisfazer as preferências de professores e alunos, reduzir o trabalho gasto durante o processo de alocação, visando respeitar as restrições e diretrizes definidas pela UNIRIO. Entretanto, aborda a geração da grade horária sem se basear ou mapear as suas respectivas salas e possui dependência com o SWI-Prolog, utilizado como mecanismo de busca para as soluções. Adicionalmente, o sistema não permite a realização de mudanças manuais na grade gerada.

Outro sistema para resolução do problema programação de horário é o Kairós [4], desenvolvido para o contexto da Faculdade Ruy Barbosa (Salvador, Bahia). O Kairós se baseia nos Algoritmos Genéticos como técnica para realizar a programação. Segundo os autores, o sistema gera soluções próximas ao ideal e permite a edição dos resultados obtidos, a fim de melhorar a solução, respeitando a todas as restrições estipuladas pelos usuários. A arquitetura *desktop*, a configuração fixa dos parâmetros do AG e o acoplamento do mecanismo de otimização com a camada de negócio, são as principais limitações do trabalho.

Em [18] uma ferramenta baseada em AGs é apresentada para automatizar o processo de alocação de salas na Universidade Federal de Uberlândia. Os resultados indicados no trabalho mostram que estratégias baseadas em AGs são viáveis para a resolução do problema de alocação de salas. Entretanto, o sistema foi desenvolvido apenas para testar a viabilidade do modelo, não disponibilizando interfaces de interação com o usuário. Sendo assim, não foi mencionado pelos autores, se houve adoção da ferramenta por parte dos responsáveis da instituição pelo processo de alocação.

O Sistema de Programação de Horários e Alocação de Salas (Schdlr), é um *software open source*, aplicado no curso de Ciência da Computação do IME/UERJ [19]. Esta solução visa resolver os problemas de programação de horário e alocação de salas, simultaneamente, utilizando das técnicas meta-heurísticas *Simulated Annealing* (SA) e *Late Acceptance Hill Climbing* (LAHC) [22]. Os resultados apresentados no trabalho demonstraram que os dois algoritmos apresentam bons resultados com o custo e tempo satisfatórios em relação à solução manual. Entretanto, o trabalho possui alguns fatores limitantes, tais como, não permitir a adaptação manual dos resultados obtidos e não possuir interfaces gráficas integradas com mecanismo que executa os algoritmos. Dada esta característica, os usuários precisam exportar os dados em formato JSON e então acionar o mecanismo motor que irá executar a otimização. Ao final da execução, o resultado é exportado em formato CSV, sendo necessário outro programa para visualizá-lo.

Tabela I. FERRAMENTAS QUE RESOLVEM PROBLEMAS DE ALOCAÇÃO DE RECURSOS EM INSTITUIÇÕES DE ENSINO

| Característica | Grade Horária [2] | Kairós [4] | Prado et al. [18] | Schdlr [19] | S.A.R.A. |
|---|---|---|---|---|---|
| Problema abordado | Grade horária | Grade horária | Alocação de salas | Grade horária e alocação de salas | Alocação de salas |
| Técnica da IES | Manual | Manual | Manual | Manual | Manual |
| Técnica aplicada | SWI-Prolog | AG | AG | SA e LAHC | AG |
| Permite alterar solução | Não | Sim | Sim | Não | Sim |
| Aplicação | Web | Desktop | Desktop | Web e Desktop | Web |
| *Open Source* | Não | Sim | Não informado | Sim | Sim |

Por fim, vale ressaltar que todas as IES que foram utilizadas como caso de uso dos trabalhos citados, utilizavam do método manual para resolver os problemas de alocação de recurso, conforme mostra a Tabela I. E, embora tais trabalhos estejam disponíveis, existem desafios associados à natureza do problema (diretrizes e normas acadêmicas que cada IES possui), que impossibilita uma solução universal a ser adotada em todos os casos [10].

## IV. A SOLUÇÃO PROPOSTA

A S.A.R.A.[1] (sara-project.me), ferramenta apresentada nesta seção, foi desenvolvida em dois sub-sistemas independentes, são eles: **sara-web** e **sara-core**. Através da combinação destes dois sub-sistemas, é possível gerenciar os dados das alocações, realizar novos mapeamentos e analisar as informações geradas, facilitando a tomada de decisão por parte dos gestores.

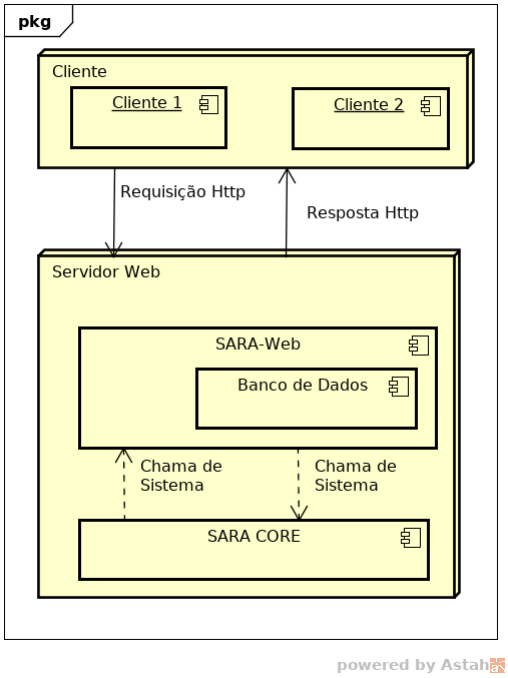


Figura 4. Diagrama de Implantação do S.A.R.A..

A Figura 4 mostra como estes módulos se relacionam e demonstra a forma como são implementados. As informações sobre os sub-sistemas supracitados serão descritas detalhadamente a seguir.

[1]S.A.R.A. compõe um projeto de software livre com licença LGPL 3.0. e seu código fonte pode ser encontrado no endereço: https://github.com/sara-project.

### A. *sara-web*

O sara-web é uma plataforma Web, baseado no estilo arquitetural Cliente-Servidor, desenvolvido utilizando o *framework* Django[2]. Dessa forma, o sistema segue o padrão arquitetural do próprio *framework*, denominado Model-View-Template (MVT) [23], que, na prática é uma nomenclatura diferente para o padrão arquitetural Model-View-Controller (MVC).

Este sub-sistema é a parte do S.A.R.A. responsável pela inserção e gestão dos dados dos modelos que fazem parte do problema de alocação de salas. A Figura 14 (ver Anexo B) apresenta o diagrama de classes dos modelos envolvidos na regra de negócio em questão e a forma com que se relacionam.

Este sistema foi desenvolvido com o objetivo de realizar a comunicação com o sara-core e facilitar a gestão das informações. A Figura 15 do Anexo B apresenta os casos de uso do sara-web.

### B. *sara-core*

O sara-core é a parte do S.A.R.A. responsável por executar a alocação de salas utilizando como estratégia de otimização os Algoritmos Genéticos. O mecanismo, desenvolvido utilizando Java 8[3], possui uma arquitetura local, porém extensível e configurável.

A arquitetura utilizada permite a mudança nas estruturas e configurações do sistema em tempo de execução através do uso de *plug-ins*. Alguns aspectos de projetos foram adotados na infraestrutura da otimização proposta, em função da complexidade elevada do problema, mesmo com instâncias de pequeno porte [6]. A Figura 5 mostra a arquitetura do sara-core, através do diagrama de componentes.

O sara-core utiliza de técnicas *multithreading*, ampliando a exploração dos recursos computacionais disponíveis a fim de otimizar o processamento das informações. Atualmente, a função de gerar novos indivíduos e a função responsável por avaliá-lo são beneficiadas pelo uso dessa técnica.

O sara-core executa as otimizações com base nos dados recebidos, mediante as solicitações enviadas pelo sara-web (neste caso). A Figura 6 ilustra o funcionamento da comunicação entre os sub-sistemas da S.A.R.A..

Conforme mostra a Figura 6, ao ser solicitada pelo usuário, a requisição é enviada para o servidor do sara-web, que por sua vez estrutura os dados envolvidos no problema em um arquivo JSON (Detalhes do arquivo no Anexo B). Em seguida,

[2]Mais informações no site oficial do projeto: djangoproject.com

[3]Site oficial do projeto: java.com

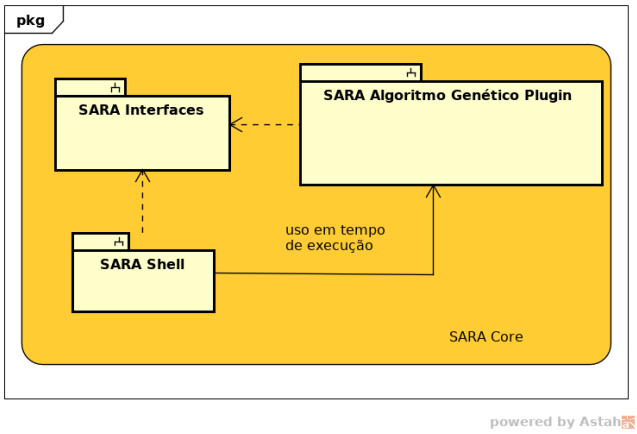


Figura 5. Diagrama de Componentes do sara-core.

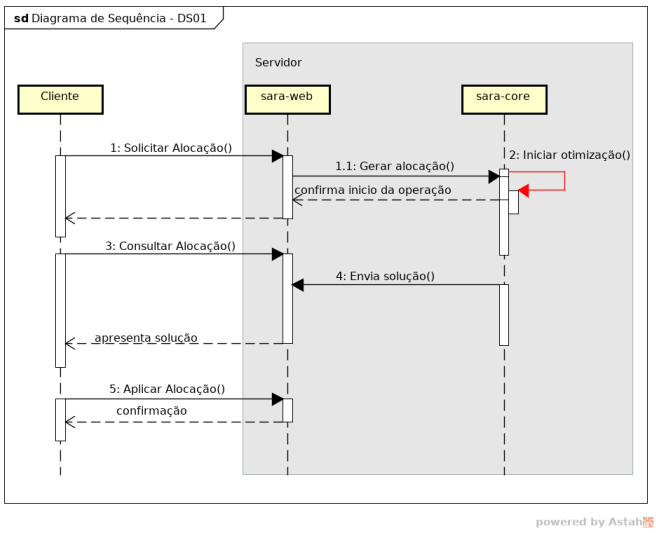


Figura 6. Diagrama de Sequencia - solicitação de nova alocação.

o sara-web realiza uma chamada para o sara-core através de chamadas de sistema, recebendo uma confirmação positiva caso a execução tenha iniciado com sucesso, caso contrário, retorna o erro para o sara-web.

A execução do sara-core é feita de forma assíncrona, não bloqueando o fluxo do solicitante. O sara-web por sua vez, poderá enviar solicitações para consultar o estado atual da alocação.

Após a realização do processo de otimização, o sara-core deixará disponível a melhor solução encontrada para o problema, bem como seus respectivos metadados, também em um arquivo JSON. Detalhes da estrutura deste arquivo é apresentado no Anexo B-C.

Vale ressaltar que o sara-core foi projetado para ser independente do sara-web. É possível acoplar o mecanismo em qualquer sistema de gestão acadêmica, desde que sejam implementadas as APIs disponibilizadas.

### C. *Descrição do Problema*

As IES, de maneira geral, apresentam um ambiente altamente dinâmico. A cada semestre letivo os dados podem sofrer diversas alterações, novos métodos e/ou sistemas são implementados. Eventualmente, no decorrer do semestre, algumas alterações podem ser demandadas, como por exemplo, a alteração no corpo docente, em decorrência de afastamentos ou término de contratos.

Especificamente, o *Campus* Salvador do IFBA possui 405 professores e 6.755 alunos, sendo 73 deles portadores de necessidades especiais. Estes dados foram obtidos do Relatório de Gestão Institucional de 2015 [24]. Tais alunos estão distribuídos entre os 26 cursos ofertados no *campus* com diferentes níveis e modalidades. Maiores detalhes são apresentados na Tabela VIII, apresentada no Anexo A.

Além disso o *campus* conta com a estrutura física de 19 blocos que abrigam salas de aula com diferentes tamanhos e características, tais como acessibilidade, equipamentos multimídia e outros recursos. No total são 52 salas de aula, 39 salas administrativas, 30 laboratórios, 4 oficinas, dentre outros espaços.

As aulas, no geral, ocorrem nos turnos matutino, vespertino e noturno, divididas em 50 minutos cada hora aula, com início às 07:00 e término às 22:00, totalizando 18 horários diários de aula, sendo 6 para cada turno. A Tabela VII ilustra como estão distribuídos os horários de aula entre os turnos, apresentando seus respectivos períodos de início e fim.

No IFBA, a grade horária de cada curso é gerenciada pelo seu coordenador. Entretanto, essa grade não pode ser criada separadamente pelos coordenadores pois os professores podem estar alocados em uma turma de diferentes departamentos. Dessa forma, a grade horária para cada semestre do IFBA é estabelecida através de uma comissão formada por coordenadores, chefes de departamento e representantes da Diretoria de Ensino (DE).

A alocação de salas só é feita após a construção das grades horárias dos cursos. O setor do IFBA responsável pelas distribuições das turmas em salas de aula utiliza um sistema Web [25] como ferramenta de apoio. Entretanto, o processo continua sendo manual, a ferramenta apenas auxilia na disposição, navegação e exibição dos dados. Além disso, a ferramenta não é normatizada e se baseia apenas em descrições textuais, tornando mais suscetível a inserção de erros e a perda da integridade da informação.

Alguns requisitos, tais como o de acessibilidade, nem sempre são levados em consideração durante o processo de alocação, muitas vezes devido a indisponibilidade desse tipo de informação no momento em que é feita a alocação das salas. Consequentemente, isso pode gerar soluções que causam transtorno aos discentes portadores de necessidades especiais.

Para que haja mais igualdade entre o atendimentos aos alunos, é fundamental designar salas devidamente equipadas às turmas que possuam alunos com necessidades especiais. Do mesmo modo, aulas que demandam equipamentos específicos, tais como laboratórios de aulas práticas, devem ser direcionadas às salas que possuam tais equipamentos.

Com base nessas informações, é possível perceber um elevado grau de complexidade atribuído ao cenário do IFBA, o que corrobora a necessidade do suporte computacional para a resolução de problemas de alocação de recurso. Além disso, otimizar este processo implica, inclusive na economia de recursos da instituição e consequentemente a redução de gastos financeiros.

### D. Restrições do Problema

Existem restrições que são comuns em qualquer IES. Contudo, o grande número de regras específicas de cada IES, baseadas em suas diretrizes, dificulta o processo de alocação de propósito geral, inviabilizando muitas vezes um mecanismo genérico de qualidade [10].

Na resolução de problemas de alocação de salas, geralmente, busca-se satisfazer os requisitos do problema, os quais são categorizados com fortes e fracas. A solução buscada deve apresentar um escalonamento das aulas que vise atender de forma efetiva e factível as restrições impostas.

Restrições fortes são caracterizadas por moldar o espaço de busca do problema, devido sua natureza obrigatória. Logo, uma solução que viole ao menos uma destas, é considerada inválida.

Para o problema estudado nesse trabalho, fazem parte das restrições fortes:

- C1 Não pode haver mais de uma aula alocada simultaneamente na mesma sala;
- C2 As aulas de uma turma não podem ser alocadas em mais de uma sala em um mesmo horário;
- C3 Não pode haver uma aula alocada a uma sala que não tenha capacidade para comportar os alunos;
- C4 Todas as turmas devem ser alocadas em salas de acordo com o seus respectivos cronogramas de aulas.

Restrições fracas são as especificações e preferências desejadas. Sendo assim, uma solução que não satisfaça um ou mais dessas, não se torna inválida, porém tem sua qualidade afetada negativamente. Dessa forma, a função objetivo deve minimizar as violações deste tipo de especificações.

As restrições fracas utilizadas no problema estudado, são:

- Q1 Todas as aulas devem ser alocadas;
- Q2 Não deve ser alocado uma turma que contenha algum aluno com mobilidade reduzida numa sala que não o comporte;
- Q3 As aulas devem ser alocadas preferencialmente em salas que possuam o mesmo tamanho da turma;
- Q4 Minimizar a troca de uma mesma turma entre blocos diferentes;
- Q5 Minimizar a troca de uma mesma turma entre salas diferentes;
- Q6 As aulas que necessitem de um tipo específico de sala devem ser alocadas nas salas com seu respectivo tipo;
- Q7 Aulas que necessitam de salas com determinado tipo de equipamento (ou recurso), devem ser alocadas em salas que o forneça.

### E. Modelagem do Problema

O sara-core lida com modelos relacionados ao dialeto dos Algoritmos Genéticos, enquanto o sara-web com o problema de alocação. Dessa forma, esta subseção tem como objetivo apresentar os elementos envolvidos, bem como elucidar sua função no problema em questão. A Figura 7 apresenta os modelos utilizados pelo S.A.R.A. sob as duas óticas: a imagem da esquerda contém os termos relacionados ao problema de alocação de salas, enquanto a imagem da direita os termos relacionados à estrutura do Algoritmo Genético.

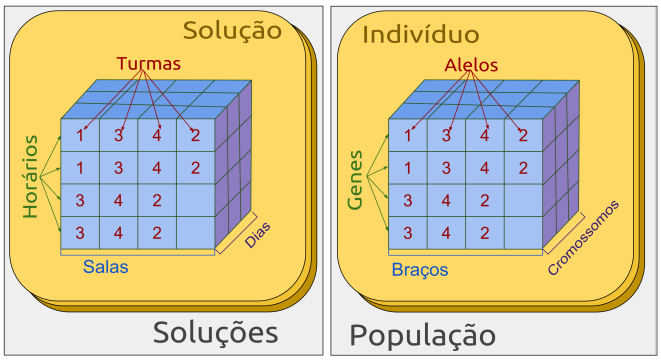


Figura 7. Representação de uma solução.

O modelo *dia* é a representação de um dia da semana (e.g. "Segunda"), enquanto o modelo *intervalo de tempo* representa uma hora aula (e.g. "17:00 - 17:50"). Dessa forma, o *cronograma* é combinação entre um *dia* e um *intervalo de tempo* (e.g. "Segunda, 17:00 - 17:50").

O modelo *sala* (e.g. "Sala 1") representa um espaço que pode ser ocupado pela *turma* (e.g. "Turma A"), que por sua vez representa um grupo de uma determinada disciplina. Uma *turma* pode conter uma ou mais *aulas* (e.g. "Aula 1, Turma A"), representadas pelo elemento *cronograma*.

A alocação se refere a uma *aula* de uma *turma* em um determinado momento da *sala*. Dessa forma, a representação desse dado momento em uma *sala* é representado pelo modelo *slot* (e.g. "Sala 1, Segunda, 17:00 - 17:50"). Este elemento é resultante da combinação única entre uma *sala* e um *cronograma*.

O total de elementos do conjunto de *slots* $S$ é resultante da multiplicação da quantidade de elementos presentes nos conjuntos *salas* $Sa$ e *cronogramas* $C$. A Equação 1 ilustra esta definição.

$$|S| = |Sa| \times |C| \tag{1}$$

Entretanto, visto que o número de elementos a serem combinados e avaliados é um fator crucial em relação ao tempo computacional utilizado, o sara-core só considera os *slots* que possuam os mesmos *cronogramas* das *aulas*. Com isso, o número de elementos *slots* é reduzido consideravelmente evitando o gasto de memória, pois os dados não vão precisar ser armazenados nem computados, além do tempo de execução, uma vez que evitará testar soluções que já estariam inválidas.

No cenário do AG, um conjunto de *slots* do mesmo *dia* forma um *cromossomo* (e.g. Slots da Segunda), que por sua

vez é formado por um conjunto de *genes*, neste caso um *slot* específico. Um *gene* contém o *alelo* que pode assumir o valor de um identificador de uma *turma* (e.g. Cromossomo 1, Gene 1, Alelo A). Caso o *alelo* não possua um valor, significa que aquele *slot* está vazio.

Como forma de facilitar as operações de busca e inserção, os *genes* foram agrupados por *salas*, formando os *braços* de um *cromossomo*. Ao obter um *braço* de um determinado *cromossomo* é possível acessar todos os *slots* de um determinada *sala* e *dia*, variando apenas o *intervalo de tempo*. (e.g. Slots da Sala 1, na Segunda).

### F. Definição da População Inicial

Uma população inicial de qualidade faz toda diferença nos resultados finais do AG [17]. Entretanto, é preciso ponderar a complexidade de gerar uma nova solução, pois isso implica no gasto de tempo e eventualmente numa solução que converge para um ótimo local mais facilmente.

Nesta etapa são alocados $x$ *aulas* em $y$ *slots* de forma aleatória. O algorítimo irá alocar uma *aula* no primeiro *slot* que o couber, independente do espaço livre que irá deixar. Entretanto, a fim de evitar a criação de *indivíduos* iguais, as listas, tanto das *aulas* como dos *slots* são embaralhadas.

Conforme dito na Subseção IV-E, a busca dos *slots* é feita baseada em dados das *aulas* (e.g. Obter todos *slots* do mesmo *cronograma* da Aula). Dessa forma, é garantida a criação de indivíduos válidos, respeitando os critérios C2 e C4. Além disso a velocidade da busca é otimizada, uma vez que o espaço de busca é reduzido, além de ser evitado gerar soluções inviáveis pela regra de incompatibilidade entre os *cronogramas* dos envolvidos.

Como estratégia para resolver o caso das *aulas* em sequência, caso o mecanismo consiga alocar uma *aula* em uma determinada *sala*, ele irá tentar alocar as aulas seguintes no mesmo espaço (caso o *slot* esteja vazio). Com isso, as soluções geradas tendem a evitar o particionamento das aulas e consequentemente mudanças de salas entre as aulas.

Além disso, cada *slot* contém uma *turma*, que equivale ao espaço que pode ser ocupado ou não por uma Turma. Sendo assim, a regra de um *slot* não conter mais de uma turma é resolvida, não sendo necessário avaliar este critério (C1). Esse procedimento se repete até que os $p$ indivíduos da população inicial sejam criados, sendo $p$ um valor definido através de um parâmetro.

A estratégia adotada para gerar uma solução inicial se assemelha com a heurística First Fit. Essa heurística é conhecida por ser rápida e consumir poucos recursos em comparação à outros métodos, tal como o *Best Fit*. Além disso, ao utilizar essa estratégia, o critério C3 não é violado, garantindo que todas soluções geradas sejam válidas, ao menos neste critério.

### G. Função de Avaliação das Soluções

Um indivíduo $s$ é avaliado por uma função $f$, denominada função de aptidão (*fitness*). A função de aptidão adotada no sara-core é representada pela Equação 2, a qual deve ser maximizada.

$$\max\ f(s) = \sum_{i=1}^{|Ch|} C(ch_i) \tag{2}$$

sendo:

$$C(ch) = \sum_{i=1}^{|Cr|} P(cr_i) \times cr_i(ch) \tag{3}$$

O conjunto $Ch$ representa os cromossomos do indivíduo $s$, sendo $ch_i$ um cromossomo desse conjunto. Cada um desses cromossomos é avaliado separadamente pela função $C(ch)$ (Equação 3).

A função $C(ch)$ irá avaliar cada cromossomo segundo o conjunto de critérios $Cr$ retornando um valor no intervalo de 0 a 1. Um critério $cr_i$ tem seu peso definido pela função $P(cr_i)$. O valor do peso estará no intervalo de 1 até o $|Cr|$. Caso este critério seja um critério forte e seu valor seja 0, a função $f$ terá seu valor igual a 0, invalidando a solução.

Conforme dito no inicio da seção, geralmente não se sabe o valor máximo que uma solução pode obter. Entretanto, no sara-core é possível saber, devido forma com que a função de aptidão foi formulada. Desde que o indivíduo tenha todos seus cromossomos atendendo 100% cada um dos critérios, o valor máximo da solução pode ser calculado pela Soma de Gauus ilustrada na Equação 4, multiplicada com o número de cromossomos.

$$S = [n(n+1)/2] \tag{4}$$

### H. Operadores Genéticos

O sara-core implementa um total de 7 operadores genéticos, sendo 2 de seleção, 3 de cruzamento e 2 de mutação. Como configuração padrão, a cada geração é escolhida de forma aleatória uma combinação desses operadores. Com isso, espera-se obter uma maior variedade da população devido a forma diferenciada que trabalha cada um dos operadores.

A seguir são listados os operadores genéticos implementados na infraestrutura do algoritmo genético no sara-core:

- Seleção por posicionamento: ao chegar nesta etapa, todos os indivíduos já estarão com sua aptidão definida, logo este operador irá ordena-lós em ordem decrescente selecionando uma porcentagem dos primeiros indivíduos.
- Seleção por torneio: dois indivíduos são selecionados aleatoriamente, o com maior aptidão é selecionado e o ciclo se repete até que uma parcela pré-definida esteja completa.
- Cruzamento de único ponto: os indivíduos selecionados no operador de seleção são escolhidos 2 a 2 e um valor de corte é escolhido aleatoriamente. A partir disso são criados dois novos indivíduos (descontentes), um com os cromossomos que sejam menor ou igual ao ponto de corte do Genitor A e os cromossomos do Genitor B que sejam maiores do que o ponto de corte. O inverso acontece para criação de outro descontente.

- Cruzamento dos melhores pontos: um descendente copia os melhores cromossomos dos seus genitores e o outro indivíduo com os demais cromossomos, com piores aptidões. Este operador foi desenvolvido particularmente para o sara-core.
- Cruzamento uniforme: parecido com cruzamento de único corte, entretanto é baseado em um vetor de inteiros com tamanho do número de cromossomos dos genitores. Este vetor é gerado de forma aleatória com 0 ou 1. Dessa forma são criados os novos descendentes, um associando o valor 0 aos cromossomos do Genitor A e 1 para o Genitor B e o outro desentende o inverso. A Figura 16 (ver Anexo B) representa o funcionamento deste operador.
- Mutação por troca em um único ponto: É escolhido de forma aleatória dois pontos quaisquer em um cromossomo, e seus valores são trocados, respeitando a regra dos cronogramas. A Figura 1 (ver Anexo B) representa o funcionamento deste operador.
- Mutação por troca em múltiplos pontos: Parecido com o operador anterior, entretanto, são selecionados múltiplos pontos do cromossomo. A Figura 18 (ver Anexo B) representa o funcionamento deste operador.

## V. Testes e Validação

Para realização dos testes, foi realizado o cadastro manual da base de dados inicial do sistema, ou seja, dos espaços físicos da instituição e de todas as aulas do curso Tecnologia em Análise e Desenvolvimento de Sistemas do *Campus* Salvador. Isto foi necessário devido a indisponibilidade da base de dados para leitura destas informações e pelo próprio processo burocrático e de Segurança da Informação de alguns setores da instituição.

Por se tratar de um pequeno recorte do cenário da instituição, foi possível obter as informações, manualmente, através de documentos disponibilizados pela coordenação do curso e através da ferramenta utilizada pelos alunos para obter informações sobre as aulas.

Além disto, também por conta da indisponibilidade dos dados, reduzimos o escopo dos testes para que considerasse apenas um dos cursos superiores desta IES. Com o intuito de investigar os benefícios da automatização do processo de alocação de salas de aula através do mecanismo proposto, foi criado um modelo que representa as aulas, salas e a alocação do curso de ADS (2017.1). A Tabela IX, apresentada no Anexo B por razões de espaço, apresenta os dados deste modelo.

Foram criadas quatro variações deste cenário, apresentadas na Tabela II, com o objetivo de simular diversas situações e analisar o comportamento do mecanismo frente a estas possibilidades. As dimensões de cada cenário criado são apresentadas na Tabela III.

Após execução e análise dos primeiros testes foi possível observar que a restrição de acessibilidade (ver Seção IV-D), apesar de ter maior relevância, ocorreu em menor frequência no cenário utilizado se comparada ao particionamento das salas. Por esta razão, estas duas restrições foram reordenadas de forma que pudessem refletir a prática institucional. A nova ordenação dos critérios é apresentada na Tabela IV, juntamente com o peso associado a cada um destes critérios.

Tabela II. Quadro ilustrativo dos cenários e suas respectivas análises, utilizados no estudo.

| Cenário | Descrição | Análise |
|---|---|---|
| C0 | Mapeamento utilizado pelo curso de ADS no semestre letivo de 2017.1. | Analisar o quão bom é este mapeamento, segundo os critérios definidos. |
| C1 | Todas as aulas de ADS de 2017.1 utilizando todas as salas que foram reservadas para pelo menos uma aula do curso. | Analisar uma disposição de aulas em salas diferente da utilizada e que apresente resultados melhores. |
| C2 | Todas as aulas de ADS de 2017.1 utilizando o mínimo necessário de salas para atender as aulas. | Analisar o comportamento do mecanismo quando há um número limitado de salas disponíveis para aulas. |
| C3 | Todas as aulas de ADS de 2017.1 utilizando um número menor de salas do que o necessário para alocar todas as aulas. | Analisar o comportamento do mecanismo quando não há um número de salas suficiente em relação ao número de aulas a serem alocadas. |
| C4 | Todas as aulas de ADS contendo 50% das turmas com alunos que precisam de acessibilidade nas salas em que forem alocadas suas aulas. | Analisar o comportamento do mecanismo quando há um volume maior de requisições por acessibilidade, visto que o cenário atual apenas 10% das aulas possuem esse requisito. |

Tabela III. Dimensão dos elementos em cada cenário.

| Informações sobre os cenários | | | | | | |
|---|---|---|---|---|---|---|
| **Cenário** | **Slots** | **Salas** | **Turmas** | **Aulas** | **Requisitos Q2** | **Requisitos Q6** |
| C0 | 980 | 35 | 36 | 156 | 18 | 82 |
| C1 | 980 | 35 | 36 | 156 | 18 | 82 |
| C2 | 504 | 18 | 36 | 156 | 18 | 82 |
| C3 | 420 | 15 | 36 | 156 | 18 | 82 |
| C4 | 980 | 35 | 36 | 156 | 78 | 82 |

Tabela IV. Critérios e pesos referentes a qualidade de uma solução.

| Peso | Descrição | Identificador (IV-D) | Tipo |
|---|---|---|---|
| 6 | Aulas não alocadas | Q1 | Fraco |
| 5 | Particionamento das salas | Q3 e C3 | Ambos |
| 4 | Tipo específico de requisitos | Q6 e Q7 | Fracos |
| 3 | Taxa da troca de salas entre aulas | Q4 e Q5 | Fracos |
| 2 | Acessibilidade | Q2 | Fraco |
| 1 | Alocação duplicada | C2 | Forte |

Ao todo são computados seis critérios, totalizando para cada cromossomo uma nota máxima igual a vinte e um (21). Como os cenários utilizados nos experimentos são baseados no curso de ADS, o qual possui aulas em cinco (5) dias da semana (cromossomos), a nota máxima que uma solução pode atingir (solução ótima global) é igual a 105 (ver Seção IV-G).

Algumas regras foram unificadas (ver Tabela IV) com o objetivo de economizar recurso computacional, uma vez que as informações são obtidas do mesmo método. Outro ponto a se observar é que, apesar de ser uma restrição forte, o C2 ficou com peso menor. Isto ocorreu porque para que uma solução ser válida, ela precisa atender a este requisito. O critério Q7 apesar de implementado (junto ao Q6), não afeta na nota final gerada, devido a indisponibilidade de informações dos recursos providos pelas salas, bem como as requisições destes recursos pelas turmas, necessárias para o cálculo.

Uma fase do experimento foi conduzida utilizando uma análise fatorial completa [26]. Este método sugere testar cada possível combinação entre todos os níveis identificados. Sendo assim é possível avaliar os efeitos que cada fator tem sobre os outros.

Este experimento é conduzido da seguinte forma:

- Gerar a quantidade $q$ de combinações possíveis, sendo $q = (x)(y)(n)$, onde $x$ representa o número de níveis do fator 1, $y$ do fator 2 e $n$ níveis do fator $N$;
- Enumerar todas as combinações que podem existir entre fatores;
- Gerar uma sequência aleatória para cada combinação;
- Realizar os experimentos seguindo a ordem obtida.

Com base nas definições apresentadas, foram conduzidos experimentos para avaliar o impacto que a variação dos valores dos parâmetros pode exercer na qualidade das soluções e no tempo de execução. A Figura 8 apresenta os parâmetros e seus respectivos valores, utilizados na execução deste ensaio. Ao todo foram 32 conjuntos de configurações possíveis a serem testadas, representados na Figura 19 (Anexo C).

**Configurações**

| Parâmetro | Valores | |
|---|---|---|
| Tamanho da População | **100** | **1000** |
| Máximo de Gerações | **100** | **1000** |
| Seleção | **25%** | **50%** |
| Mutação | **5%** | **10%** |
| Elitismo | **0%** | **10%** |

Figura 8. Parâmetros do Algoritmo Genético e seus possíveis valores.

O experimento baseado na análise fatorial completa levou em consideração apenas o cenário C1, devido a alta demanda de tempo requisitado para realização deste experimento. A execução de todos os conjuntos de parâmetros para o cenário C1 resultou em 11 horas e 30 minutos de duração (aproximadamente). A Figura 9 apresenta o tempo de execução (em ordem crescente) de cada combinação.

Dado o fator do tempo, foi decidido selecionar um subconjunto dessas combinações, sendo 3 com melhores resultados, ilustrados na Figura 10 e 3 com os resultados mais baixos, apresentados na Figura 11.

A Figura 24 apresenta (em ordem crescente) as melhores soluções encontradas de cada configuração testada no cenário 1, enquanto a Figura 27 ilustra as melhores soluções encontradas nos demais cenários em relação as configurações selecionadas.

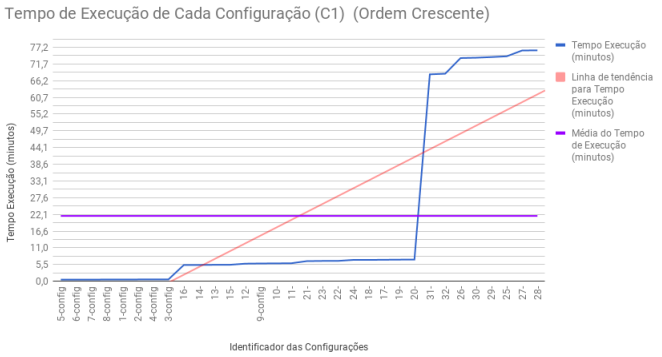


Figura 9. Tempos de execução (em ordem crescente) para cada configuração (apresentadas na Figura 19).

**Melhores Soluções**

| Configuração | População | Geração | Seleção | Mutação | Elitismo |
|---|---|---|---|---|---|
| 24-config | 1000 | 100 | 50% | 10% | 10% |
| 27-config | | 1000 | 25% | | |
| 23-config | | 100 | | | 0% |
| 32-config | | 1000 | 50% | | |
| 18-config | | 100 | 25% | 5% | 10% |

Figura 10. Configurações que geraram as melhores soluções para o cenário 1.

**Piores Soluções**

| Configuração | População | Geração | Seleção | Mutação | Elitismo |
|---|---|---|---|---|---|
| 6-config | 100 | 100 | 50% | 5% | 10% |
| 4-config | | | 25% | 10% | |
| 1-config | | | 25% | 5% | 0% |
| 9-config | | 1000 | 25% | | |
| 14-config | | | 50% | | 10% |

Figura 11. Configurações que geraram as piores soluções para o cenário 1.

Apesar da grande demanda em termos de tempo de execução, dois testes foram executados com o objetivo de avaliar a ordem de grandeza do tempo de execução em função dois parâmetros. Um trata-se crescimento do tamanho da população, representado pela Figura 12, e o outro baseado no número máximo de gerações, representado pela Figura 13. Conforme mostram as Figuras citadas, houve pouca variação no total do tempo de execução.

Para que fosse possível realizar uma comparação viável entre as soluções geradas pela ferramenta proposta e o método atual utilizado, foi aplicada a mesma função de avaliação na solução atual do curso de ADS (C0). Entretanto, como a solução atual é feita manualmente, alguns critérios são violados, como por exemplo alocar uma turma de 45 alunos numa sala com 40 lugares, uma vez que já se considera histórico de evasões e transferências entre instituições e cursos, o que é relativamente frequente entre alunos do primeiro semestre.

Este tipo de violação não ocorre no sara-core e portanto,

Tabela V. QUADRO COMPARATIVO ENTRE OS DIVERSOS CENÁRIOS ESTUDADOS.

| Cenário | Configuração | Nota da solução | Tempo de execução (minutos) | Aulas não alocadas | Lugares Vazios nas Salas | Aulas que não atendem Q6 | Aulas que não atendem Q2 |
|---|---|---|---|---|---|---|---|
| C0 | — | 0 | Indefinido | 0 | 338 | 18 | 2 |
| C0 (Flexibilizado) | — | 73,046646 | Indefinido | 46 | 338 | 28 | — |
| C1 | 24-Config | 82,879684 | 7,07 | 0 | 756 | 56 | 0 |
| C2 | 23-Config | 82,71032 | 5,33 | 6 | 862 | 36 | 0 |
| C3 | 24-Config | 78,99315 | 4,87 | 32 | 864 | 36 | 0 |
| C4 | 27-Config | 82,906845 | 70,81 | 0 | 716 | 64 | 0 |

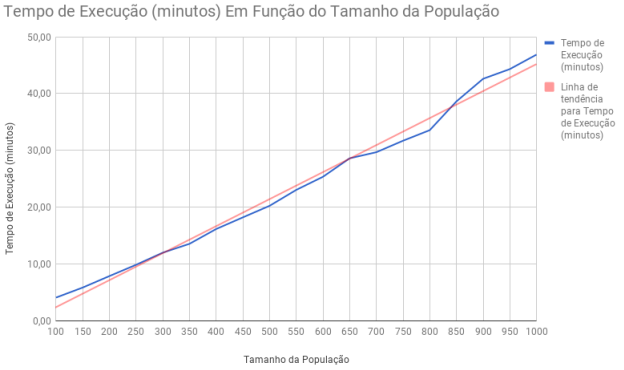


Figura 12. Ordem de grandeza do tempo de execução em função da variação do tamanho da população.

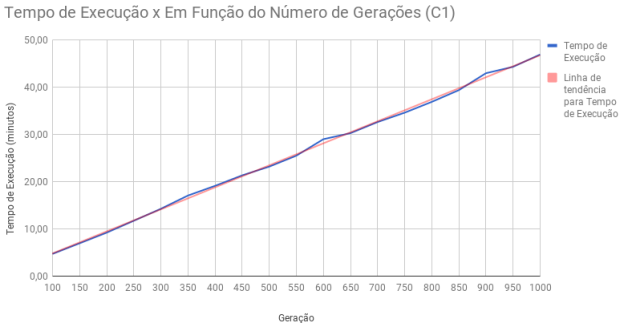


Figura 13. Ordem de grandeza do tempo de execução em função da variação do número máximo de gerações.

estas duas alocações não podem ser diretamente comparadas. A Tabela VI aponta as salas que tiveram essa ocorrência, as turmas que estão alocadas à elas e o número da sobrecarga de alunos.

Vale ressaltar que o tamanho das turmas se refere a quantidade de vagas ofertadas e não quantidade de alunos matriculados. A implementação seguiu esta premissa porque estes foram os dados disponibilizados pela instituição. Logo, as soluções poderiam representar melhor a realidade do uso das salas, caso fossem utilizados os números de alunos matriculados em cada turma.

Dessa forma, foi decidido realizar uma nova avaliação, desta vez porém, sem alocar as turmas que invalidam a solução. Sendo assim uma nota pode ser atribuída à solução, porém gerando um número maior de aulas não alocadas, critério este que tem peso maior da qualidade final da solução.

Conforme mostra a Tabela V, as soluções geradas pela S.A.R.A. apresentam maiores lugares vazios nas salas e alguns casos mais aulas que não atendem critério Q6. Isto ocorre pois muitas aulas que requisitavam por Q6 são de turma com tamanhos maiores do que o tamanho dos laboratórios.

Além disso a área dos laboratórios, até dado momento, não possui acessibilidade. A ferramenta buscou atender o critério definido como mais importante, o número máximo de aulas alocadas. Por este motivo os cenários C1, C2, C3 e C4 apresentaram número maior de lugares vazios.

Entretanto, é possível perceber que a S.A.R.A. se adaptou bem ao cenários criados, gerando soluções mais relevantes, quando comparado a solução atual, sem ter um crescimento abrupto do tempo de execução. Em contrapartida, o método adotado atualmente no IFBA, demanda vários dias e recursos humanos para criar um escalonamento das aulas, ocorrendo muitas vezes, mudanças com o semestre letivo em andamento.

O tempo de processamento é referente aos valores dos parâmetros e o *hardware* em que a ferramenta for executada. Por este motivo, devem ser comparados apenas quando executados em ambientes com a mesma configuração. Os experimentos apresentados nesta seção foram executados em um computador com processador Intel® Core™ i5-6200U 2.30GHz e 8GB de memória RAM e sistema operacional Linux 4.14.14-1-Manjaro.

## VI. CONCLUSÃO

Atualmente, o processo de designar turmas em salas de aula no IFBA é executado de maneira manual. Até então, não foi encontrada uma ferramenta disponível que pudesse ser utilizada adequadamente para minimizar o tempo requerido e os esforços empreendidos pelas pessoas que realizam esta tarefa.

O subsistema sara-web, cujo objetivo é a exibição e manipulação dos dados, foi implementado com foco na facilidade de gerenciamento das informações. Já o sara-core, responsável pela otimização do processo de alocação de salas, o foco foi na velocidade de execução e eficiência do método adotado. Através da modelagem do problema e da configuração dos elementos e operadores genéticos desenvolvidos foi possível encontrar uma maneira de tratar as especificidades do problema no contexto do IFBA.

Tabela VI. TURMAS E SALAS AFETADAS COM A SOBRECARGA, UTILIZADAS NO MAPEAMENTO DE AULAS DO CENÁRIO DE ADS (2017.1).

| | Salas | Capacidade | Turma | Aulas | Tamanho | Sobrecarga |
|---|---|---|---|---|---|---|
| 1 | O-118 | 30 | MAT222 - T01 | 2 | 50 | 20 |
| 2 | L-105 | 40 | LET100 - T04 | 4 | 50 | 10 |
| | | | ADM500 - T03 | 2 | 48 | 8 |
| 2 | LAB 03 | 40 | INF026 - T01 | 2 | 50 | 10 |
| 3 | LAB 01 | 20 | INF031 - T01 | 2 | 30 | 10 |
| | | | INF017 - T01 | 2 | 40 | 20 |
| | | | INF018 - T01 | 4 | 40 | 20 |
| | | | INF016 - T01 | 6 | 40 | 20 |
| | | | INF011 - T01 | 2 | 40 | 20 |
| 4 | B-209 | 40 | HUM102 - T01 | 2 | 60 | 20 |
| 5 | B-107 | 45 | HUM102 - T01 | 2 | 60 | 15 |
| 6 | O-207 | 40 | ADM500 - T02 | 2 | 48 | 8 |
| 7 | D-102 | 35 | INF010 - T01 | 2 | 40 | 5 |
| 8 | D-102 | | HUM100 - T01 | 2 | 50 | 15 |
| 9 | D-111 | 35 | HUM100 - T01 | 2 | 50 | 15 |
| 10 | K-115 | 20 | INF023 - T01 | 2 | 40 | 20 |
| | K-115 | 20 | ADM545 - T01 | 2 | 40 | 20 |
| 11 | D-105 | 35 | INF023 - T02 | 2 | 40 | 5 |
| 12 | K-110 | 20 | ADM545 - T01 | 2 | 40 | 20 |
| **Totais** | **12** | **575** | **16** | **46** | **856** | **281** |

Ainda, para validação do resultado, as solução encontradas a partir das execuções do sara-core, foram comparadas aos resultados obtidos com a solução atualmente utilizada (alocação manual). A comparação se deu baseada no critério do tempo de execução e no atendimento das restrições estabelecidas.

Considerando os resultados obtidos, é possível afirmar que estratégias baseadas em Algoritmos Genéticos podem ser uma boa abordagem para a resolução do problema de alocação de salas. O desempenho apresentado pelo mecanismo, bem como a disponibilidade de soluções de boa qualidade a um baixo custo computacional e a forma adaptável da sua infraestrutura ao problema em questão, corroboram com esta afirmação.

Tendo em vista que o processo de mapeamento das aulas é executado uma vez a cada início de semestre, o resultado da alocação não precisa necessariamente ser de imediato. Isto é, mesmo que uma solução computacional requeira alguns minutos de execução, ainda assim será mais eficiente, e potencialmente menos suscetível a erros do que uma abordagem baseada na alocação manual, principalmente se a quantidade de elementos a serem relacionados for relativamente grande, como é o caso do problema tratado aqui.

### A. *Trabalhos Futuros*

Durante o desenvolvimento do projeto, surgiram diversas possibilidades de implementação com o escopo para trabalhos futuros. E, após a execução dos experimentos foi possível evidenciar mais possibilidades de extensão da ferramenta. Visando evolução do projeto S.A.R.A., destacam-se como trabalhos futuros:

- Atualmente, no sara-web, só é possível solicitar uma nova alocação para todos os modelos cadastrados. Sendo assim, é necessário implementar uma forma de selecionar um grupo específico de turmas, salas e horários específicos para serem alocados;
- Disponibilizar um parâmetro que permita atribuir uma taxa máxima de convergência das soluções, modificando algo no comportamento do sistema como uma estratégia para sair dos ótimos locais;
- Dotar a ferramenta com a opção de atribuir uma taxa de flexibilidade para cada turma em relação ao tamanho das salas. Essa permitirá que a alocação não seja binária (pode ou não pode);
- Parametrizar a configuração dos critérios. Dessa forma será possível alterar ou remover requisitos de preferências de acordo com o cenário atual;
- Visto que o problema de alocação de salas geralmente é de natureza multiobjetivo (maximizar qualidade e minimizar conflitos), dotar o mecanismo com técnicas de Otimização Multiobjetivo, se mostra uma abordagem promissora;
- Alterar o sara-core para uma arquitetura distribuída, tal como Master/Slave. Com isso será possível trabalhar com instâncias maiores e mais complexas do problema;
- Disponibilizar uma API, bem como uma documentação de uso e implantação, para que outras instituições possam se beneficiar do uso desta ferramenta. Apenas sendo necessário adaptar as especificações e ao seu contexto;
- Permitir a parametrização do término da otimização baseado em um tempo, ou estimativa da qualidade;
- Disponibilizar um módulo de gestão e configuração do sara-core, permitindo modificar os parâmetros,

operadores e estratégias utilizadas pelo sistema, tanto *off-line* quanto *on-line*. Além disso, resultados e detalhes do processo em andamento através de interface gráfica, dando um melhor *feedback* da otimização;

- Ampliar os aspectos de análise da S.A.R.A., podendo avaliar o tempo médio da otimização baseada na instância do problema. Além disso, permitir a exibição do tempo previsto para para o término da execução, bem como dados da melhor solução encontrada, atualizados em tempo real.

## REFERÊNCIAS


[1] O. de Oliveira Braz Júnior, "Otimização de Horários em Instituições de Ensino Superior Através de Algoritmos Genéticos," Master's thesis, Universidade Federal de Santa Catarina, Florianópolis, 2 2000.

[2] D. R. D. Berardino and A. L. Correa, "Software para Geração da Grade Horária do curso BSI da UNIRIO," 2013, monografia (Bacharel em Sistemas da Informação), UNIRIO (Universidade Federal do Estado do Rio de Janeiro), RJ, Brasil.

[3] V. S. Dani and G. Bernardi, "Desenvolvimento de Interface Web Multiusuário para Sistema de Geração Automática de Quadros de Horários Escolares," 2014, monografia (Bacharel em Ciência da Computação), UFSM (Universidade Federal de Santa Maria), RS, Brasil.

[4] C. C. Freitas, P. R. B. Guimarães, M. C. M. Neto, and F. J. R. Barboza, "Uma Ferramenta Baseada em Algoritmos Genéticos para a Geração de Tabela de Horário Escolar," *Sétima Escola Regional de Computação Bahia-Sergipe, Vitória da Conquista*, 2007.

[5] A. Schaerf, "A survey of automated timetabling," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 87–127, 1999.

[6] H. C. de Almeida Soares and L. A. A. Meira, "Um estudo de Caso sobre o Problema de Alocação," 2011, universidade Federal de São Paulo, São José dos Campos - SP.

[7] A. Subramanian, J. M. Medeiros, L. Cabral, and M. Souza, "Aplicação da metaheurística busca tabu ao problema de alocação de aulas a salas em uma instituição universitária," *Revista Produção Online*, vol. 11, no. 1, 2011.

[8] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, vol. 12, no. 1, pp. 55–89, Feb 2009.

[9] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[10] C. D. L. Hamawaki and K. Yamanaka, "Geração Automática de Grade Horária Usando Algoritmos Genéticos: O Caso da Faculdade de Engenharia Elétrica da UFU," Master's thesis, Universidade Federal de Uberlândia, 2005.

[11] P. L. B. de Souza and C. T. Scarpin, "Aplicação de algoritmos genéticos e métodos evolucionários na resolução do problema de alocação de turmas: Caso ufpr," *Blucher Marine Engineering Proceedings*, vol. 1, no. 1, pp. 625 – 636, 2014.

[12] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[13] J. Becceneri, F. Manuel Ramos, H. Campos Velho, J. Demisio Simões da Silva, L. Antonio Nogueira Lorena, N. Vijaykumar, R. Santos, R. Rosa, and J. dos Santos Travelho, "Meta-heurísticas e otimização combinatória: Aplicações em problemas ambientais," 01 2008.

[14] R. Kripka and M. Kripka, "Simulated annealing aplicado na optimização da alocação de salas em instituição de ensino superior," 09 2017.

[15] C. R. Reeves, Ed., *Modern Heuristic Techniques for Combinatorial Problems*. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[16] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[17] F. Bueno, "Métodos Heurísticos: Teoria e Implementações," 2009, tutorial, IFSC/Araranguá.

[18] G. P. Theodoro, I. S. Peretta, and K. Yamanaka, "Utilização de algoritmos genéticos para o problema de alocação de salas da universidade federal de uberlândia," *XIII Encontro Nacional de Inteligência Artificial e Computacional*, 2016.

[19] D. H. da Silva Costa, I. M. Coelho, and P. E. D. Pinto, "Programação de Horários e Alocação de Salas de Aula no IME/UERJ com Simulated Annealing e LAHC," 2017, monografia (Bacharel em Ciência da Computação), UERJ (Universidade do Estado do Rio de Janeiro), RJ, Brasil.

[20] A. S. Prado and S. R. de Souza, "Problema de alocação de salas em cursos universitários: Um estudo de caso," *XLVI Simpósio Brasileiro de Pesquisa Operacional*, 09 2014.

[21] P. Ceccon, D. Vianna, C. Bazilio, and B. Diniz, "Heurísticas iterated local search e guided local search aplicadas na resolução do problema de alocação de salas," *XVIII Simpósio de Engenharia de Produção*, 11 2011.

[22] E. K. Burke and Y. Bykov, "The late acceptance hill-climbing heuristic," *European Journal of Operational Research*, vol. 258, no. 1, pp. 70 – 78, 2017.

[23] T. D. Book. (2017) The model-view-controller design pattern. [Online]. Available: https://djangobook.com/model-view-controller-design-pattern/

[24] (2015) Relatório de gestão institucional: Ifba 2015. [Online]. Available: https://portal.ifba.edu.br/proap/transparencia-arquivos/relatorios-de-gestao-do-ifba/2015-relatorio-de-gestao-institucional

[25] (2000) Mrbs: Introduction. [Online]. Available: http://mrbs.sourceforge.net/

[26] R. Jain, *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.*, ser. Wiley professional computing. Wiley, 1991.

# ANEXO A
# DADOS DO CENÁRIO DO IFBA, *Campus* SALVADOR

Tabela VII. HORÁRIOS DE AULA DO IFBA, *Campus* SALVADOR, DIVIDIDOS EM PERÍODOS DE INÍCIO E FIM.

| Turno | Matutino | | | | | | Vespertino | | | | | | Noturno | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| Início | 07:00 | 07:50 | 08:40 | 9:30 | 10:20 | 11:10 | 12:00 | 12:50 | 13:40 | 14:30 | 15:20 | 16:10 | 17:00 | 17:50 | 18:40 | 19:30 | 20:20 | 21:10 |
| Fim | 07:50 | 08:40 | 09:30 | 10:20 | 11:10 | 12:00 | 12:50 | 13:40 | 14:30 | 15:20 | 16:10 | 17:00 | 17:50 | 18:40 | 19:30 | 20:20 | 21:10 | 22:00 |

Tabela VIII. CURSOS OFERTADOS PELO IFBA, *Campus* SALVADOR

| | Nível | Sub-categoria | Cursos |
|---|---|---|---|
| 1 | Técnico | Integrado | Automação Industrial |
| 2 | | | Edificações |
| 3 | | | Eletrônica |
| 4 | | | Eletrotécnica |
| 5 | | | Geologia |
| 6 | | | Mecânica |
| 7 | | | Química |
| 8 | | | Refrigeração e Climatização |
| 9 | | Subsequente | Automação e Controle Industrial |
| 10 | | | Eletrotécnica |
| 11 | | | Hospedagem |
| 12 | | | Téc. Em Instalação e Manutenção Eletrônica |
| 13 | | | Manutenção Mecânica Industrial |
| 14 | | PROEJA | Saneamento |
| 15 | Superior | Bacharelado | Administração |
| 16 | | | Engenharia Elétrica |
| 17 | | | Engenharia Industrial Elétrica |
| 18 | | | Engenharia Industrial Mecânica |
| 19 | | | Engenharia Química |
| 20 | | Graduação Tecnológica | Análise e Desenvolvimento de Sistemas |
| 21 | | | Eventos |
| 22 | | | Tecnologia em Radiologia |
| 23 | | Licenciatura | Geografia |
| 24 | | | Matemática |
| 25 | | | Química |
| 26 | | | Física |
| 27 | Pós-Graduação | Cursos de Especialização | Ciência e Tecnologia Ambiental |
| 28 | | | Computação Distribuída e Ubíqua |
| 29 | | | Educação Profissional, Científica e Tecnológica |
| 30 | | | Estudos Étnicos e Raciais: Identidades e Representação |
| 31 | | | Gestão de Instituições Públicas de Ensino |
| 32 | | | Educação Profissional Técnica de Nível Médio Integrada ao Ensino Médio na Modalidade Educação de Jovens e Adultos |
| 33 | | | Técnico em Segurança, Meio Ambiente e Saúde |
| 34 | | Cursos de Mestrado | ProfEPT – Mestrado Profissional em Educação Profissional e Tecnológica em Rede Nacional |
| 35 | | | Ciências e Tecnologias Ambientais |
| 36 | | | Engenharia de Sistemas e Produtos |
| 37 | | | Programa de Pós-Graduação em Propriedade Intelectual e Transferência de Tecnologia para a Inovação |

Fontes: http://portal.ifba.edu.br/menu-ensinos/cursos. Acessado em 07/12/2017. http://www.prpgi.ifba.edu.br/cursos/ Acessado em 30/03/2018.

Tabela IX. TURMAS DO CURSO DE ADS DO IFBA, *Campus* SALVADOR

| **Código** | **Descrição** | **Carga Horária** | **Semestre** | **Vagas Ofertadas** | **Requisitos por Acessibilidade** | **Requisito por Laboratórios de Informática** |
|---|---|---|---|---|---|---|
| ADM500 | Introdução a Administração - T02 | 60 | 1 | 48 | | |
| | Introdução a Administração - T03 | | 1 | 48 | | |
| INF026 | Introdução a Computação | 60 | 1 | 50 | | **Sim** |
| INF027 | Introdução à Lógica - T01 | 60 | 1 | 25 | | **Sim** |
| INF027 | Introdução à Lógica - T02 | 60 | 1 | 25 | | |
| INF027 | Introdução à Lógica - T03 | 60 | 1 | 25 | | |
| LET100 | Língua Portuguesa | 60 | 1 | 50 | | |
| MAT222 | Matemática I | 90 | 1 | 50 | **Sim** | |
| HUM102 | Psicologia Aplicada ao Trabalho | 60 | - | 60 | | |
| HUM103 | Metodologia de Pesquisa | 60 | 2 | 30 | | |
| INF006 | Estrutura de Dados e Algoritmos | 90 | 2 | 40 | | **Sim** |
| INF028 | Arquitetura de Computadores e Software Básico | 60 | 2 | 40 | | |
| INF029 | Laboratório de Programação | 60 | 2 | 40 | | **Sim** |
| LET102 | Inglês - T01 | 60 | 3 | 30 | | |
| LET102 | Inglês - T02 | 60 | 3 | 30 | | |
| INF007 | Banco de Dados I | 90 | 3 | 40 | | **Sim** |
| INF009 | Sistemas Operacionais | 90 | 3 | 40 | | **Sim** |
| INF021 | Estágio Supervisionado | 300 | 3 | 40 | | |
| INF008 | Programação Orientada a Objetos | 90 | 3 | 40 | | **Sim** |
| INF011 | Padrões de Projeto | 90 | 4 | 40 | | **Sim** |
| INF012 | Programação Web | 90 | 4 | 40 | | **Sim** |
| INF010 | Banco de Dados II | 90 | 4 | 40 | **Sim** | **Sim** |
| INF015 | Rede de Computadores I | 60 | 4 | 40 | **Sim** | **Sim** |
| INF030 | Métodos Científicos em Computação | 30 | 5 | 30 | **Sim** | |
| INF014 | Engenharia de Software | 90 | 5 | 40 | | Sim |
| INF017 | Rede de Computadores II | 60 | 5 | 40 | | |
| INF022 | Tópicos Avançados | 60 | 5 | 40 | | |
| ADM550 | Empreendedorismo | 60 | 5 | 40 | | |
| HUM100 | Filosofia | 60 | 5 | 50 | | |
| INF018 | Auditoria e Segurança de Sistemas | 60 | 6 | 40 | | |
| INF019 | Gerência de Projetos | 60 | 6 | 20 | | |
| INF016 | Arquitetura de Software | 90 | 6 | 40 | | **Sim** |
| INF020 | Sistemas Distribuídos | 60 | 6 | 40 | | **Sim** |
| INF023 | Trabalho de Conclusão de Curso - T01 | 60 | 6 | 40 | | |
| INF023 | Trabalho de Conclusão de Curso - T02 | 60 | 6 | 40 | | |

# ANEXO B
# DOCUMENTAÇÃO S.A.R.A

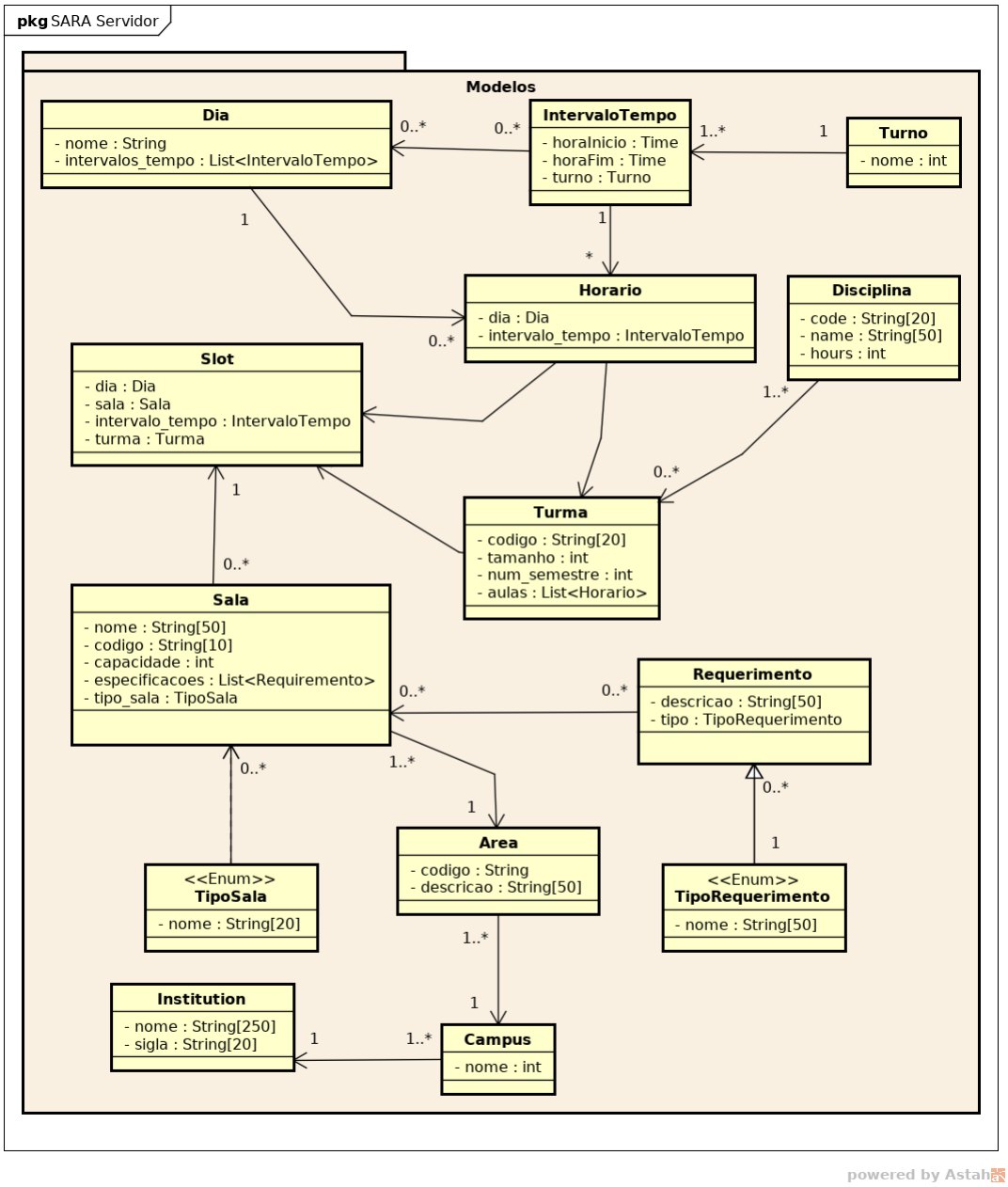


Figura 14. Diagrama de Classes que representa os modelos envolvidos no problema de alocação de sala.

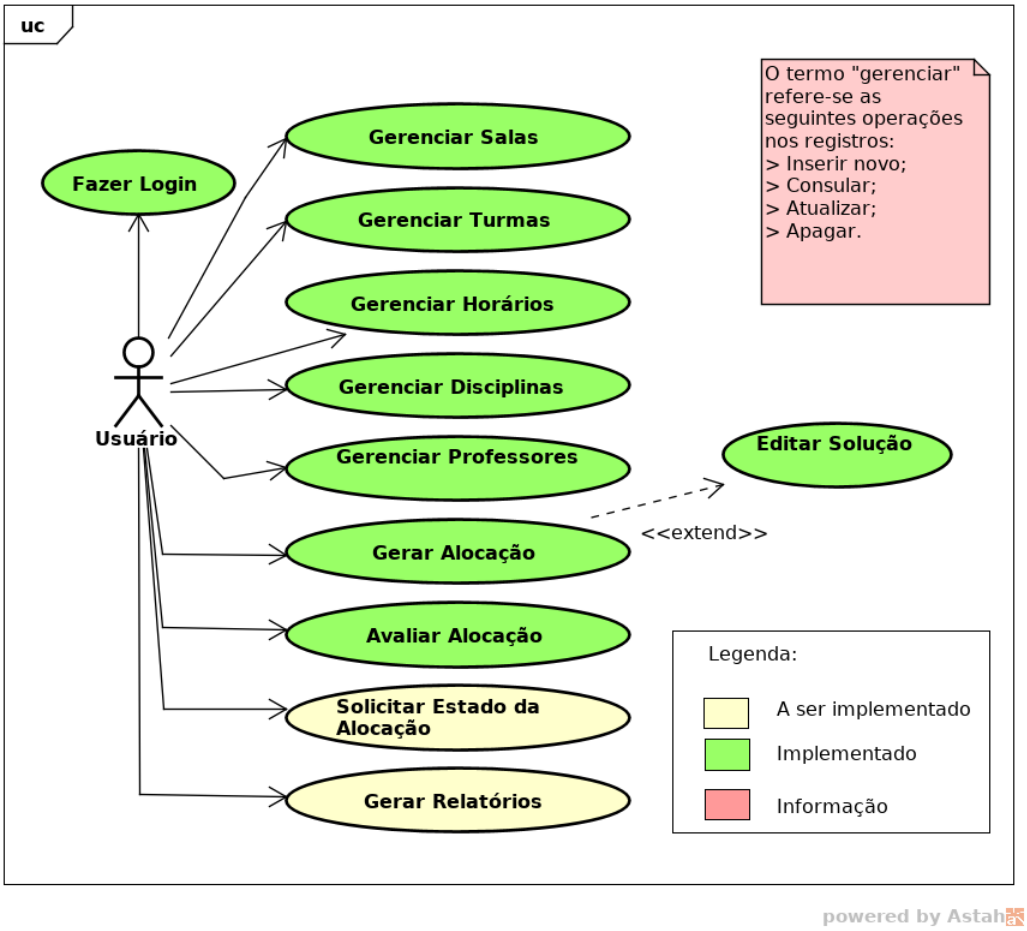


Figura 15. Caso de uso do sub-sistema sara-web.

*A. Representação dos Operadores Genéticos*

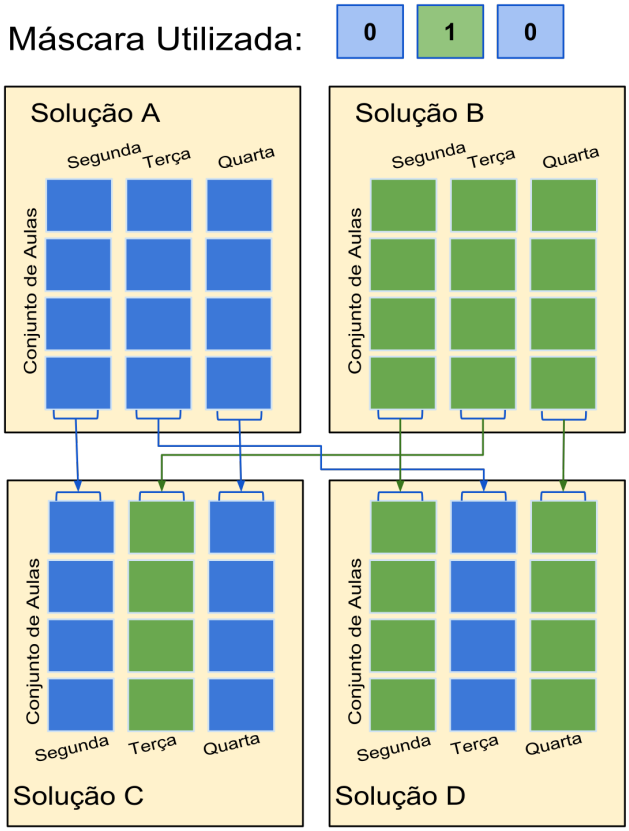


Figura 16. Representação do operador cruzamento uniforme adaptado ao problema, combinando elementos de duas soluções distintas.

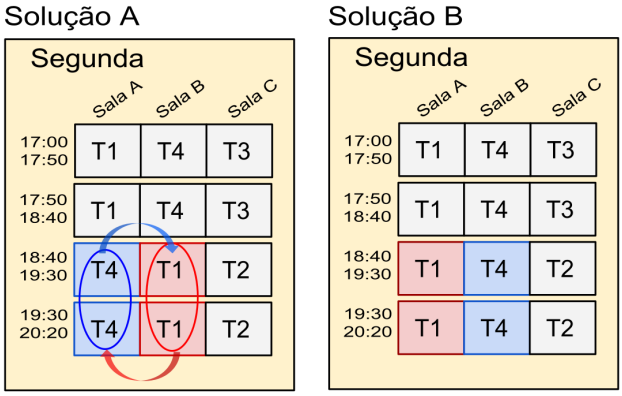


Figura 17. Representação do operador mutação por troca em um único ponto, adaptado ao problema, trocando aulas entre duas salas distintas da mesma solução.

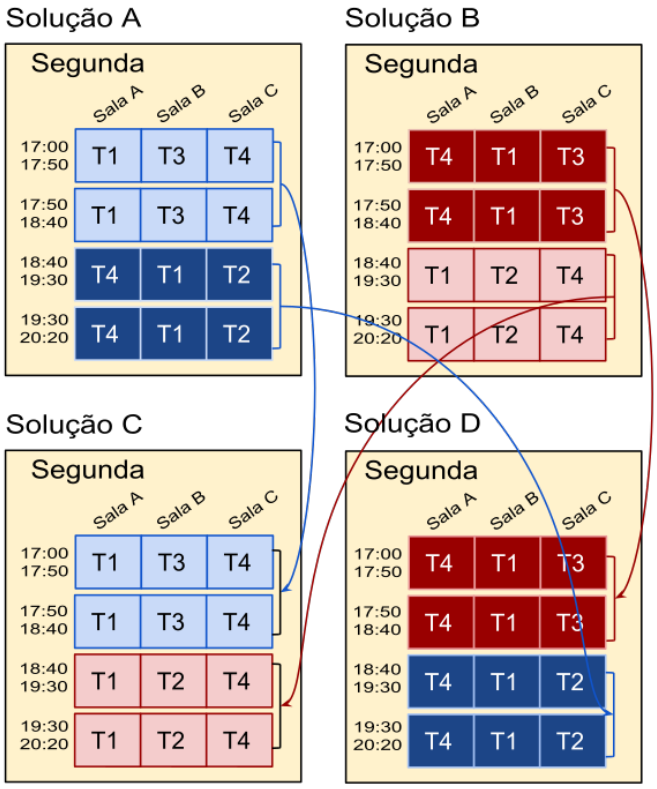


Figura 18. Representação do operador mutação por troca em múltiplos pontos, adaptado ao problema, trocando aulas entre duas salas distintas da mesma solução.

*B. Exemplo de uma requisição em formato JSON, enviada para o sara-core*

Um arquivo de requisição **deve** conter o conjunto de elementos envolvidos no processo de alocação de salas e o tipo da solicitação, sendo "class_assignment" para realizar uma nova alocação e "eval_solution" para avaliação de uma solução existente.

Além disso, este arquivo **pode** conter os valores para configuração dos parâmetros do AG. Entretanto, o sara-core possui configurações padrão para o AG, caso esta informação não esteja contida no conteúdo da requisição.

Um exemplo deste arquivo é apresentado abaixo:

```
[{
  "request_type": "class_assignment",
  "ga_config": {
        "population_number": 500,
        "max_generation": 1000,
        "mutation_probability": 0.05,
        "select_probability": 0.5,
        "elitism_probability": 0.1
  },
  "schedules": [{
    "id": 13,
    "day": 1,
    "time_interval": 14
  }],
  "requirements": [{
    "id": 1,
    "type": 1,
    "priority": 3
  }],
  "rooms": [{
    "id": 29,
    "area":2,
    "type":1,
    "specifications": [1],
    "capacity": 40
  }],
  "slots":[{
    "id":1,
    "room":29,
    "schedule":13
  }],
  "classes": [{
    "id": 37,
    "size": 48,
    "schedules": [13],
    "requirements": [1],
    "type_rooms_wanted":[1]
  }]
}]
```

*C. Exemplo de um resultado da execução do sara-core, em formato JSON*


```
[{
    "slots": [{
        "s_class": 2,
        "day": 1,
        "time_interval": 16,
        "room": 3
      },{
        "s_class": 2,
        "day": 1,
        "time_interval": 17,
        "room": 3
      }],
    "info": [{
        "1_request_type": "class_assignment",
        "2_execution_time": 100,
        "3_total_memory_used": "694,50 MB",
        "4_time_generate_initial_population": 0,
        "5_average_time_fitness": 0,
        "6_average_time_selection": 0,
        "7_average_time_crossover": 0,
        "8_average_time_mutation": 0,
        "9_average_time_refresh_population": 0,
        "10_doesnt_meets_accessibility_requirement": 0,
        "11_meets_accessibility_requirement": 2,
        "12_total_accessibility_requirement": 2,
        "13_empty_slots": 0,
        "14_filled_slots": 2,
        "15_duplicate_allocations": 0,
        "16_unused_places": 30,
        "17_overload_Rooms": 0,
        "18_unallocated_class_schedules": 0,
        "19_allocated_class_schedules": 2,
        "20_class_schedules_with_lab_requirement": 0,
        "21_class_meets_schedules_with_lab_requirement": 0,
        "22_best_solution_fitness": 81.00287,
        "23_fitness_time_line": [
          79.20441,
          80.47247,
          80.54262,
          81.00287
        ]
      }
    ]
  }
]
```

# ANEXO C
## DADOS DOS EXPERIMENTOS

| Parâmetro | #1-config | #2-config | #3-config | #4-config | #5-config | #6-config | #7-config | #8-config |
|---|---|---|---|---|---|---|---|---|
| Tamanho da População | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Máximo de Gerações | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Seleção | 25% | 25% | 25% | 25% | 50% | 50% | 50% | 50% |
| Mutação | 5% | 5% | 10% | 10% | 5% | 5% | 10% | 10% |
| Elitismo | 0% | 10% | 0% | 10% | 0% | 10% | 0% | 10% |

| Parâmetro | #9-config | #10-config | #11-config | #12-config | #13-config | #14-config | #15-config | #16-config |
|---|---|---|---|---|---|---|---|---|
| Tamanho da População | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Máximo de Gerações | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| Seleção | 25% | 25% | 25% | 25% | 50% | 50% | 50% | 50% |
| Mutação | 5% | 5% | 10% | 10% | 5% | 5% | 10% | 10% |
| Elitismo | 0% | 10% | 0% | 10% | 0% | 10% | 0% | 10% |

| Parâmetro | #17-config | #18-config | #19-config | #20-config | #21-config | #22-config | #23-config | #24-config |
|---|---|---|---|---|---|---|---|---|
| Tamanho da População | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| Máximo de Gerações | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| Seleção | 25% | 25% | 25% | 25% | 50% | 50% | 50% | 50% |
| Mutação | 5% | 5% | 10% | 10% | 5% | 5% | 10% | 10% |
| Elitismo | 0% | 10% | 0% | 10% | 0% | 10% | 0% | 10% |

| Parâmetro | #25-config | #26-config | #27-config | #28-config | #29-config | #30-config | #31-config | #32-config |
|---|---|---|---|---|---|---|---|---|
| Tamanho da População | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| Máximo de Gerações | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 |
| Seleção | 25% | 25% | 25% | 25% | 50% | 50% | 50% | 50% |
| Mutação | 5% | 5% | 10% | 10% | 5% | 5% | 10% | 10% |
| Elitismo | 0% | 10% | 0% | 10% | 0% | 10% | 0% | 10% |

Figura 19. Todas possíveis combinações de configurações dos parâmetros do AG.

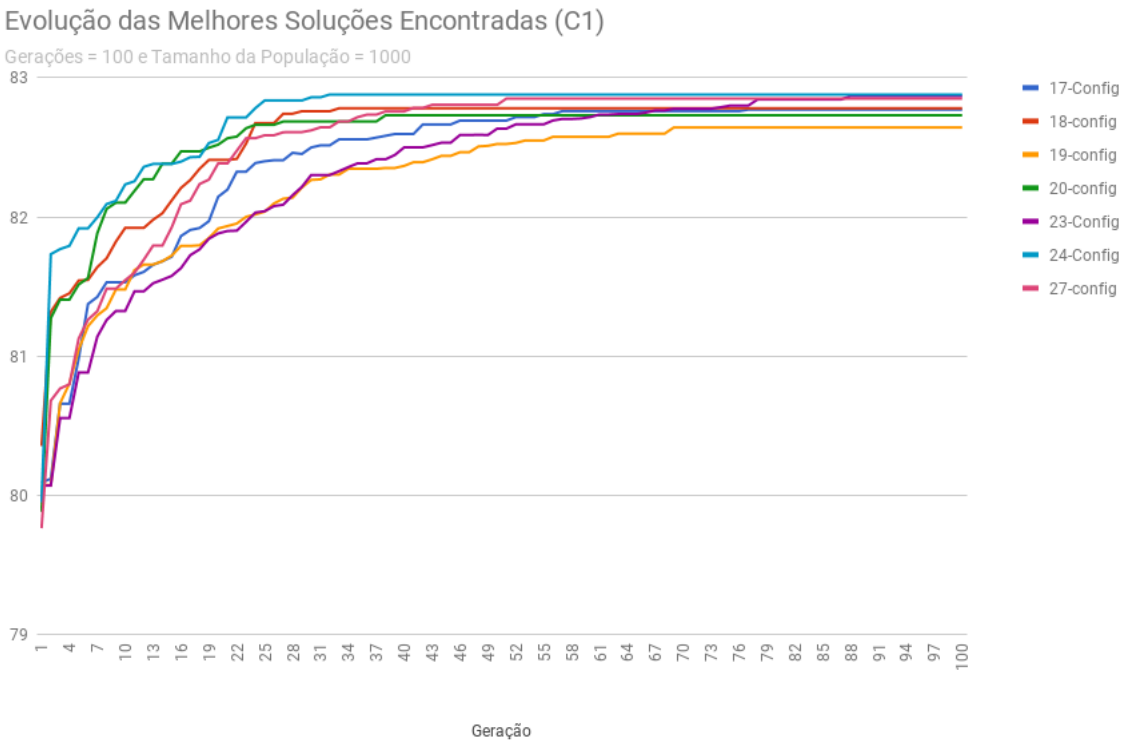


Figura 20. Evolução das melhores soluções encontradas a cada geração do cenário 1 (C1), para todas as configurações em que o número máximo de gerações é de 100.

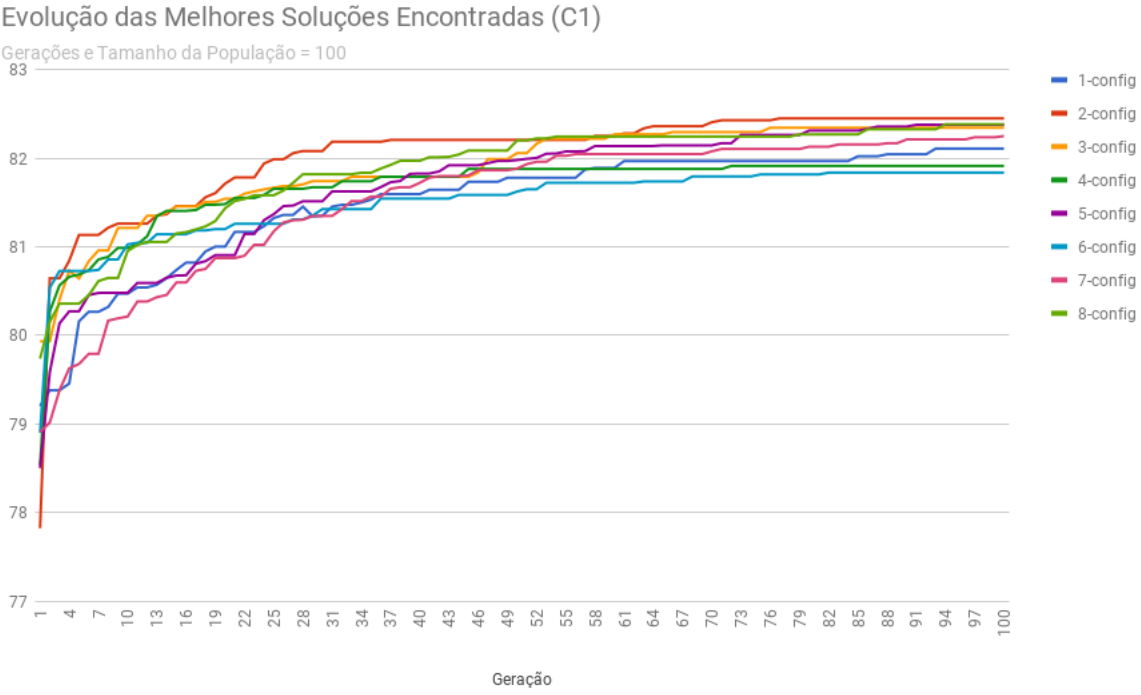


Figura 21. Evolução das melhores soluções encontradas a cada geração do cenário 1 (C1), para todas as configurações em que o número máximo de gerações é de 1000.

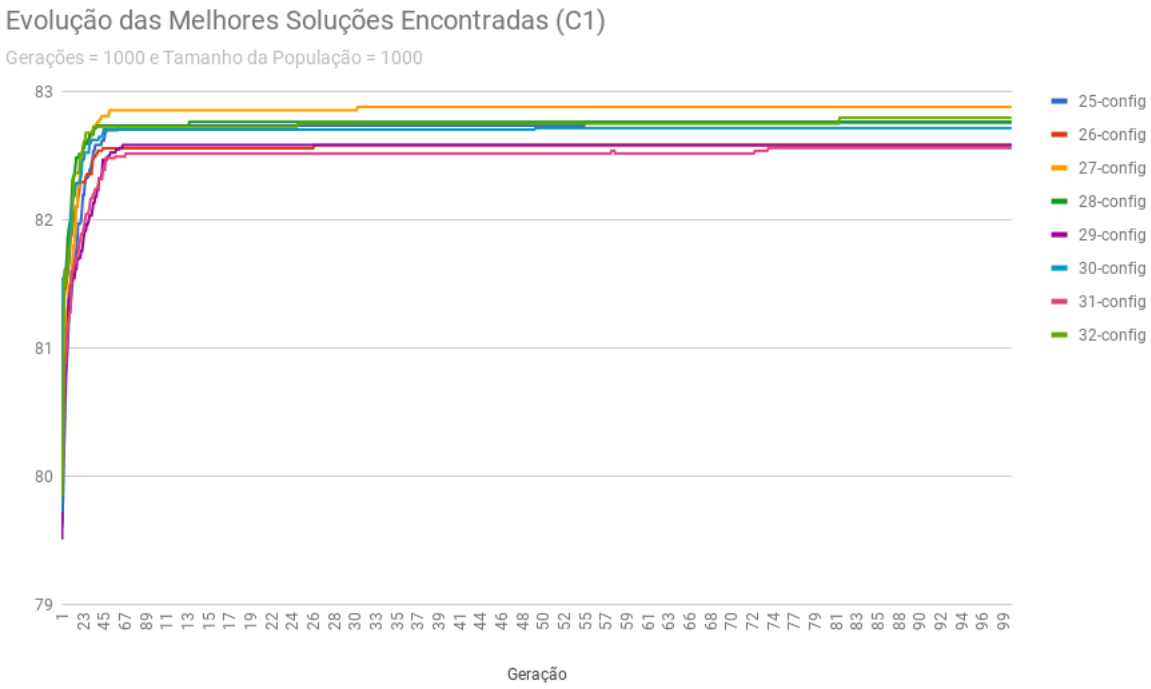


Figura 22. Evolução das melhores soluções encontradas a cada geração do cenário 1 (C1), para todas as configurações em que o número máximo de gerações é de 100.

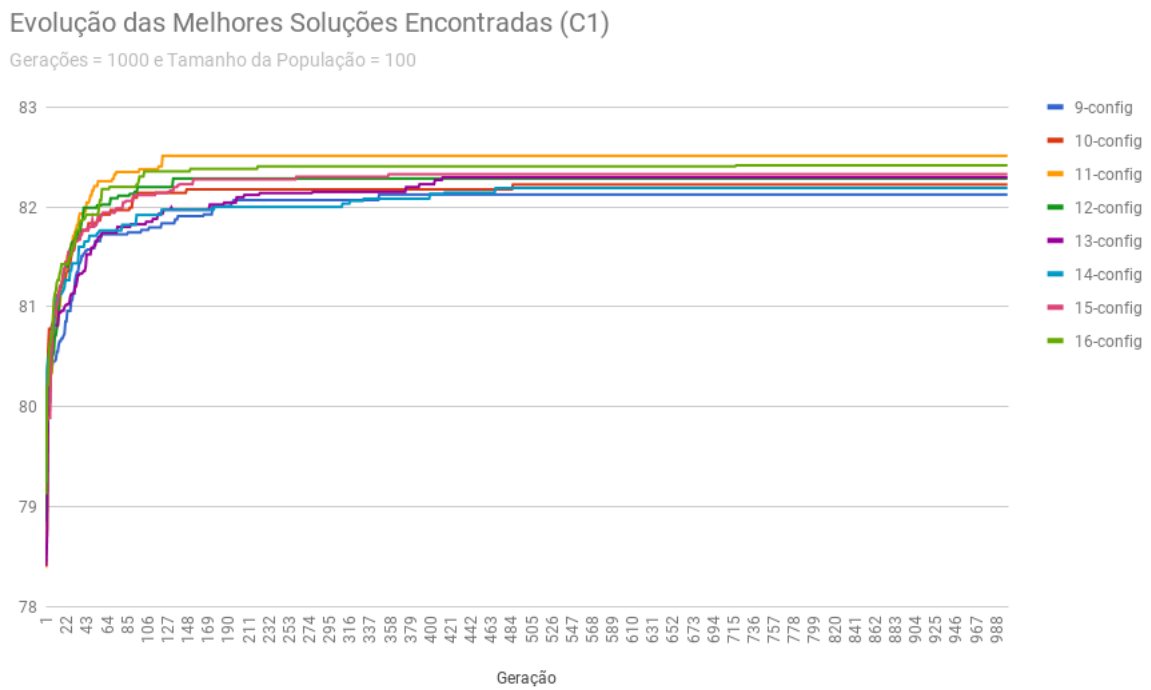


Figura 23. Evolução das melhores soluções encontradas a cada geração do cenário 1 (C1), para todas as configurações em que o número máximo de gerações é de 1000.

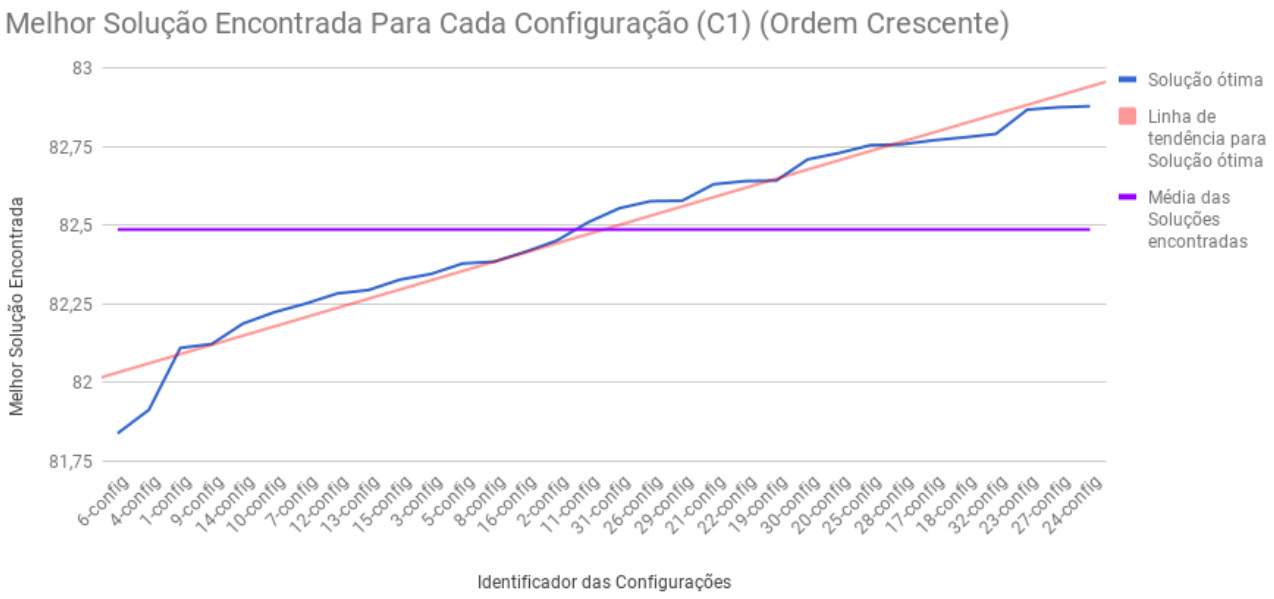


Figura 24. Melhores soluções encontradas (em ordem crescente) para cada configuração (apresentadas na Figura 19).

| Maiores Tempos de Execução | | | | | |
|---|---|---|---|---|---|
| Configuração | População | Geração | Seleção | Mutação | Elitismo |
| 28-config | 1000 | 1000 | 25% | 10% | 10% |
| 27-config | | | | | 0% |
| 25-config | | | | 5% | |
| 29-config | | | 50% | | |
| 30-config | | | | | 10% |

Figura 25. Configurações que geraram os maiores tempos de execução para o cenário 1.

| Menores Tempos de Execução | | | | | |
|---|---|---|---|---|---|
| Configuração | População | Geração | Seleção | Mutação | Elitismo |
| 5-config | 100 | 100 | 50% | 5% | 0% |
| 6-config | | | | | 10% |
| 7-config | | | | 10% | 0% |
| 8-config | | | | | 10% |
| 1-config | | | 25% | 5% | 0% |

Figura 26. Configurações que geraram os menores tempos de execução para o cenário 1.

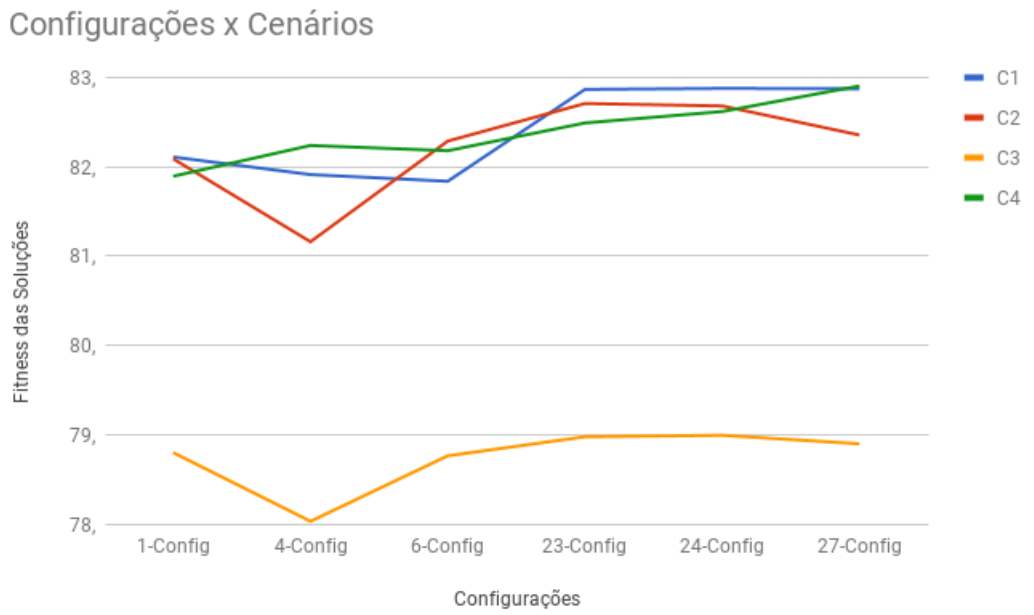


Figura 27. Melhores soluções encontradas dos cenários em relação às configurações utilizadas.

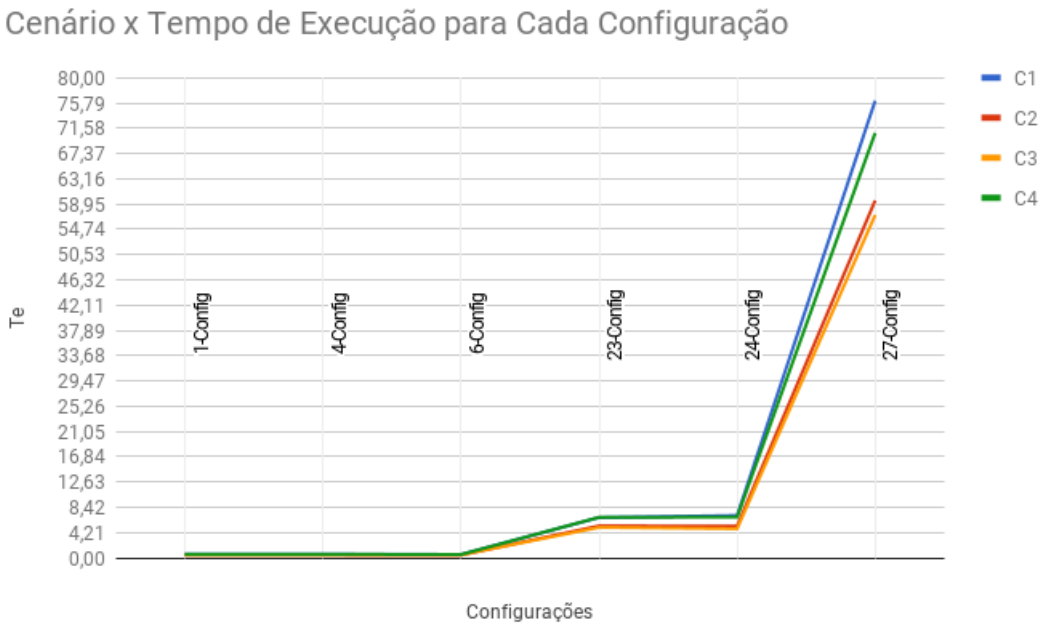


Figura 28. Tempos de execução dos cenários em relação às configurações utilizadas.

### A. *Avaliação do Mapeamento Utilizado em ADS, 2017.1*

Os códigos abaixo representam os resultados gerados na avaliação do mapeamento utilizado no curso de ADS no semestre letivo 2017.1.

Resultado gerado com a restrição que invalida uma solução caso alguma sala tenha uma turma maior do que sua capacidade:


```
[{
"best_fitness": 0,
"info": [{
    "1_request_type": "eval_solution",
    "2_execution_time": 223,
    "3_total_memory_used": "119,00 MB",
    "4_time_generate_initial_population": 23,
    "5_average_time_fitness": 52,
    "10_doesnt_meets_accessibility_requirement": 2,
    "11_meets_accessibility_requirement": 16,
    "16_unused_places": 338,
    "17_overload_Rooms": 46,
    "13_empty_slots": 824,
    "14_filled_slots": 156,
    "12_total_accessibility_requirement": 18,
    "20_class_schedules_with_lab_requirement": 82,
    "19_allocated_class_schedules": 156,
    "21_class_meets_schedules_with_lab_requirement": 64,
    "18_unallocated_class_schedules": 0,
    "22_best_solution_fitness": 0,
    "15_duplicate_allocations": 0
  }]
}]
```


Resultado gerado sem a restrição que invalida uma solução caso alguma sala tenha uma turma maior do que sua capacidade, porém não alocando estas turmas:


```
[{
"best_fitness": 73.046646,
"info": [{
    "1_request_type": "eval_solution",
    "2_execution_time": 228,
    "3_total_memory_used": "119,00 MB",
    "4_time_generate_initial_population": 23,
    "10_doesnt_meets_accessibility_requirement": 0,
    "11_meets_accessibility_requirement": 12,
    "12_total_accessibility_requirement": 14,
    "13_empty_slots": 870,
    "14_filled_slots": 110,
    "15_duplicate_allocations": 0,
    "16_unused_places": 338,
    "17_overload_Rooms": 0,
    "18_unallocated_class_schedules": 46,
    "19_allocated_class_schedules": 110,
    "20_class_schedules_with_lab_requirement": 82,
    "21_class_meets_schedules_with_lab_requirement": 54,
    "22_best_solution_fitness": 73.046646
  }]
}]
```