Estendendo a Capacidade de um Simulador de Redes de Sensores Sem Fios

Phillipe Cavalcante[1], **Evellyn Cavalcante, Tamer Cavalcante, Heitor Ramos**[2], **Alejandro C. Frery**[1]

[1] Instituto de Computação
Centro de Pesquisa em Matemática Computacional – CPMAT
Universidade Federal de Alagoas – UFAL

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais & UFAL

{cavalcante.phillipe,evellyn.cavalcante,tamergsc}@gmail.com

{heitor.ramos,acfrery}@gmail.com

Abstract. *This paper presents an extension of Sinalgo, simulator for wireless sensor networks. This extension aims at providing it with the ability of generating new scenarios of spatial distribution of nodes. This extension was developed in the R platform, which also provides other functions described in this work. After briefly reviewing the area of wireless sensor networks, we present an analysis of the structure of these platforms and how they communicate with other software entities and between themselves, a detailed description of the point processes models to be incorporated, the results achieved and future work.*

Resumo. *Este trabalho apresenta uma extensão do simulador de redes de sensores sem fios Sinalgo no sentido de dotá-lo da capacidade de gerar diversos cenários de distribuições espaciais de nós. Essa extensão foi desenvolvida na plataforma R, que provê outras funções descritas no trabalho. Para tanto, após comentar brevemente a área de redes de sensores sem fios, apresentamos uma análise da estrutura dessas duas plataformas e de como elas podem comunicar-se tanto com outras entidades de software quanto entre si, uma descrição detalhada dos modelos de processos pontuais a serem incorporados, os resultados alcançados e trabalhos futuros.*

1. Introdução

As Redes de Sensores Sem Fios (RSSFs) constituem uma das tecnologias que mais tem despertado a atenção tanto da indústria quanto da comunidade de pesquisadores dos últimos anos. Elas são formadas por conjuntos tipicamente grandes (centenas ou mais) de dispositivos autônomos capazes de (i) medir alguma grandeza do ambiente como, por exemplo, a temperatura, a iluminação, a concentração de substâncias químicas, aceleração, velocidade e número de eventos, e (ii) entrar em contato com outros dispostivos similares [Akyildiz et al. 2002, Yick et al. 2008].

A capacidade de comunicação é empregada para otimizar a autonomia do sensoriamento. Caso cada um dos sensores fosse obrigado a enviar os valores que capta para uma estação central, o custo em energia da comunicação dos dados faria com que a sua bateria se esgotasse rapidamente. Para amenizar o custo dessa comunicação, os sensores se comunicam entre si e, de acordo com algum protocolo, apenas um resumo dos dados coletados é enviado [Nakamura et al. 2007].

Um assunto considerado central para a área é a medida do desempenho de uma RSSF. A definição de desempenho depende da aplicação, e pode estar relacionada ao tempo de vida da rede, à qualidade das medições obtidas [Frery et al. 2008], à capacidade de cobertura da rede, dentre outros fatores. Uma vez identificada a métrica de desempenho de interesse, é fundamental elencar e controlar os fatores que têm incidência sobre ela.

Um dos fatores mais negligenciados na literatura é a distribuição espacial dos sensores. Os principais trabalhos da área empregam ou uma distribuição regular [Reis et al. 2007] ou uma distribuição completamente aleatória [Heinzelman et al. 2002]. A realidade, frequentemente, reside entre esses dois casos extremos.

Uma das ferramentas mais importantes nos estudos de RSSF é o simulador. Ao invés de colocar fisicamente RSSFs no terreno, o simulador é um ambiente computacional que permite realizar experiências virtuais, com as quais é possível tirar conclusões relevantes para a aplicação. Um dos simuladores mais interessantes é o Sinalgo. Ele, contudo, é limitado quanto aos modelos de distribuição espacial de sensores.

Este trabalho tem por objetivo mostrar uma extensão do Sinalgo no sentido de dotá-lo da capacidade de simular outras distribuições espaciais de sensores.

A seção 2 descreve as duas plataformas de interesse: Sinalgo e R. A seção 3 apresenta uma introdução à teoria dos processos estocásticos pontuais espaciais, o referencial teórico para a extensão aqui proposta. A seção 4 mostra resultados de utilizar a extensão do Sinalgo, e a seção 5 conclui o artigo apontando trabalhos futuros.

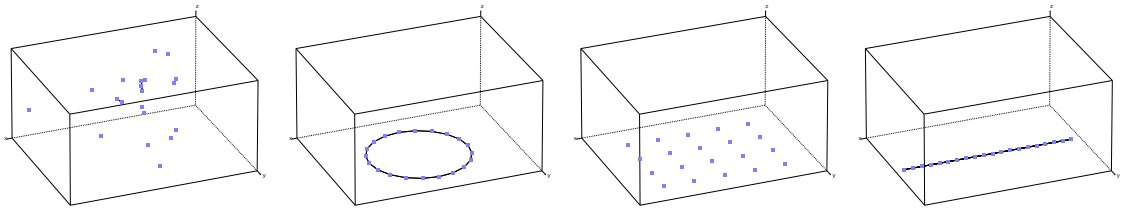
2. Sinalgo e R

Esta seção descreve brevemente as principais características das duas plataformas utilizadas neste trabalho. Fazemos ênfase nas modalidades de interoperabilidade que Sinalgo e R oferecem, por ser essa a propriedade que torna possível o desenvolvimento da extensão aqui apresentada.

2.1. O simulador Sinalgo

O Sinalgo (Simulator for Network Algorithms) é um simulador de algoritmos para redes desenvolvido em Java, cujos principais objetivos são o teste e a validação. Java é uma linguagem de programação desenvolvida pela Sun Microsystems Inc., com ampla difusão em uma grande diversidade de aplicações. Essa linguagem é atraente, dentre outros motivos, pela sua independência de plataforma, a sua modularidade e sua facilidade de codificação.

O Network Simulator 2 (NS2), simulador amplamente utilizado pela comunidade de redes, dá suporte a vários cenários de simulações. Ele leva em consideração problemas como, por exemplo, consumo de energia dos sensores e conectividade entre os nós



(a) Random (b) Circle (c) Grid2D (d) Line2D

Figura 1. Diversos cenários de topologia gerados pelo Sinalgo

sensores. Entretanto, o NS2 não consegue simular redes de grande porte. O Sinalgo, por outro lado, provê suporte a grandes redes, podendo realizar simulações com até 10^5 nós em tempo aceitável [Distributed Computing Group 2008]. Esta capacidade de simulação é alcançada pela simplificação das propriedades físicas da propagação dos sinais, ponto em que o NS2 é forte.

Uma simulação no Sinalgo tem como elementos bases nós e modelos. Os nós podem ser vistos como um modo de representação de um sensor, enquanto os modelos descrevem o ambiente na qual a rede foi simulada. Cada nó contem modelos de conectividade, de mobilidade, de interferência e de confiabilidade. Existem ainda outros modelos como, por exemplo, os modelos de distribuição e de transmissão de mensagens.

Cada modelo desempenha um papel na simulação. O modelo de conectividade decide para quais vizinhos um nó tem conexão e se encarrega de inserir e remover ligações. Já o modelo de mobilidade descreve como o nó deve se movimentar durante a simulação. O modelo de interferência é responsável pela decisão do que o nó deve fazer caso ocorra alguma interferência no envio de mensagens.

O modelo de confiabilidade decide se cada mensagem enviada deve ou não chegar ao destino. Este modelo é útil para simular perdas de pacotes na rede. O modelo de transmissão de mensagens decide o máximo alcance da mensagem, e o modelo de distribuição descreve como os nós se organizam no ambiente.

Novas simulações podem ser criadas a partir dos exemplos contidos no projeto do Sinalgo. Dentro dele há uma pasta denominada `template` que serve para iniciar uma nova simulação; nela se encontra toda a estrutura de pastas, o arquivo de configuração, dentre outros arquivos. Para criar uma nova simulação basta renomear a pasta `template` com o nome do projeto desejado e criar subclasses dentro das pastas referentes ao modelo que irá sofrer modificações.

A distribuição do Sinalgo inclui alguns modelos e permite a criação de novos. Na figura 1 é possível visualizar os quatro tipos de modelos de distribuição disponíveis: Random (figura 1(a)), Circle (figura 1(b)), Grid2D (figura 1(c)) e Line2D (figura 1(d)). O modelo Random baseia-se em 5 distribuições de probabilidade para gerar a localização dos nós no espaço: Gaussiana, Poisson, Exponencial, Uniforme e Constante. Para definir um novo modelo de distribuição é necessário uma nova classe que estenda `sinalgo.models.DistributionModel`, implementando as funcionalidades do novo modelo, dentro da pasta `models/distributionModels/`.

Na criação de novos projetos no Sinalgo pode existir a necessidade de usar fun-

cionalidades de outros sistemas. Então, ao invés de reimplementar funções de outros softwares, uma maneira de resolver esse problema é interligar o Sinalgo a esses sistemas. Para isso, deve-se analisar os modos de integração com Java. A linguagem possui sua própria plataforma, composta pelas Máquina Virtual Java (MVJ) e a Java API (Java Application Programming Interface). A MVJ é responsável pela interpretação do código intermediário gerado no processo de compilação, chamado bytecode e a Java API consiste num conjunto predefinido de classes.

Java disponibiliza algumas ferramentas para utilizar funções implementadas em outras linguagens de programação. O Java Native Interface (JNI) possibilita aos desenvolvedores aproveitarem o poder da plataforma Java, sem ter que abandonar seus investimentos em código legado, pois o JNI incorpora o código nativo escrito em linguagens como C e C++ em código escrito em Java [Liang 1999].

2.2. A plataforma R

R é um ambiente composto por vários recursos, dentre eles uma linguagem de programação (também chamada R), produção de gráficos de alto nível e interface para outras linguagens. Distribuído gratuitamente sob a licensa GNU *General Public License*, R foi especificamente construído para análise de dados estatísticos [R Development Core Team 2009]. R tem sido amplamente utilizado em várias áreas do conhecimento, principalmente pela sua precisão numérica [Almiron et al. 2009], além de fornecer funções que facilitam diversos estudos estatísticos.

R é um sistema simples executado a partir da linha de comando [R Development Core Team 2008b]. Como a linguagem R é interpretada, esta característica beneficia os usuários que não possuem afinidade com programação, pois os resultados de cada processamento podem ser verificados imediatamente, descartando a necessidade de escrever várias linhas de código para compilar e gerar um executável, como ocorre na linguagem C, por exemplo.

A linguagem R teve influência principalmente de duas outras linguagens: S e Scheme. S é também uma linguagem utilizada para análises estatísticas em cuja sintaxe e tipos de dados vetoriais R foi baseada, o que tornou-as parcialmente compatíveis (alguns *scripts* funcionam em ambas). Por outro lado, a semântica foi inspirada na programação funcional de Scheme, que é um dos dois principais dialetos de Lisp.

Uma característica forte da linguagem R é sua portabilidade, que provém, em nível de implementação, da decisão de escrevê-la em ANSI padrão C. Essa escolha se deveu ao fato de que C é a linguagem de programação padrão para muitas plataformas, além de tornar a escrita eficiente, o código compacto e facilitar o acesso a recursos de níveis mais baixo da máquina. Adicionalmente, foi utilizado um tradutor FORTRAN-to-C (`f2c` - AT&T/Bellcore) para obter fácil acesso a códigos de aplicações úteis em FORTRAN, quando necessário.

Todas as funções de R estão disponíveis por meio de pacotes. As funções básicas que formam o núcleo de R encontram-se nos pacotes chamados `base`. Funções mais complexas são fornecidas em diversos outros pacotes, que o usuário pode facilmente instalar através do próprio sistema. Além disso, a linguagem provê flexibilidade suficiente para a fácil criação de novos pacotes (em qualquer linguagem) de acordo com os requisitos de cada análise.

Uma análise estatística frequentemente acarreta na manipulação de uma grande quantidade de dados. Em muitas situações é necessário um deslocamento intenso de dados a fim de fundir funcionalidades de várias plataformas para melhorar a credibilidade do trabalho. Isso pode causar improdutividade devido ao tempo gasto com esse deslocamento. R prevê situações como essas e oferece facilidades de intercambio entre plataformas diferentes [R Development Core Team 2008a].

A maneira mais simples de realizar esse intercambio é através de arquivos textos. Seja na importação ou exportação de dados é possível utilizar recursos simples de R que proporcionam a leitura ou a criação de arquivos textos com a formatação fornecida pelo usuário e compatível com R. Essa funcionalidade seria útil, por exemplo, para realizar análises dos dados presentes em uma planilha eletrônica ou, ao contrário, carregar os resultados obtidos com a linguagem R na planilha. Para melhor viabilizar esse intercambio, é possível dispensar a atuação do usuário na formatação dos arquivos utilizando uma linguagem de marcação: eXtensible Markup Language – XML. Através dessa tecnologia a estrutura do arquivo é definida de forma transparente e automática tornando-o auto-descritivo.

R pode interagir com outros softwares estatísticos (EpiInfo, Minitab, S-PLUS, SAS, SPSS, Stata, Systat, entre outros) pela troca de arquivos binários gerados por essas plataformas. No entanto, existem cenários em que o uso desses arquivos textos ou binários não é vantajoso, por exemplo, quando é desejável incorporar código de R em outra linguagem e vice-versa. R também dá esse suporte através de funções de interface `.C`, .FORTRAN, .Call e .External [R Development Core Team 2008c]. Elas provêem uma interface para chamar funções compiladas em C ou FORTRAN a partir de R com a ressalva de que haja uma coerção explícita e correta dos tipos de dados dos argumentos de R para a linguagem estrangeira. Para fornecer a funcionalidade contrária, ou seja, chamar R a partir de C ou FORTRAN, existe uma API estável que fornece várias arquivos de cabeçalhos que podem ser incluídos no código C e alguns em Fortran. Com ela é possível inserir chamadas a funções de R no código da linguagem estrangeira.

Existem pacotes que fornecem os recursos necessários para facilitar a integração entre R e outras linguagens utilizando estes artifícios: rJava, RSPerl e RSPython.

2.3. Comunicação entre Sinalgo e R

RJava é um pacote que, utilizando as vantagens de R e Java serem interoperáveis com C, proporciona uma interface para comunicação bilateral entre essas plataformas, como vemos na figura 2.

Com essa ferramenta é possível tanto incorporar código R em Java quanto o inverso. A primeira funcionalidade é responsabilidade de apenas uma parte do pacote, a JRI (Java/R Interface), que pode ser utilizada separadamente. Neste trabalho a integração entre Sinalgo e R é feita através da JRI. Nesse processo são gerados dados utilizando o pacote `spatstat` [Baddeley and Turner 2008] do R, que produz eventos de processos estocásticos pontuais em duas dimensões, segundo uma distribuição. O Sinalgo usa estes resultados para obter topologias de redes.

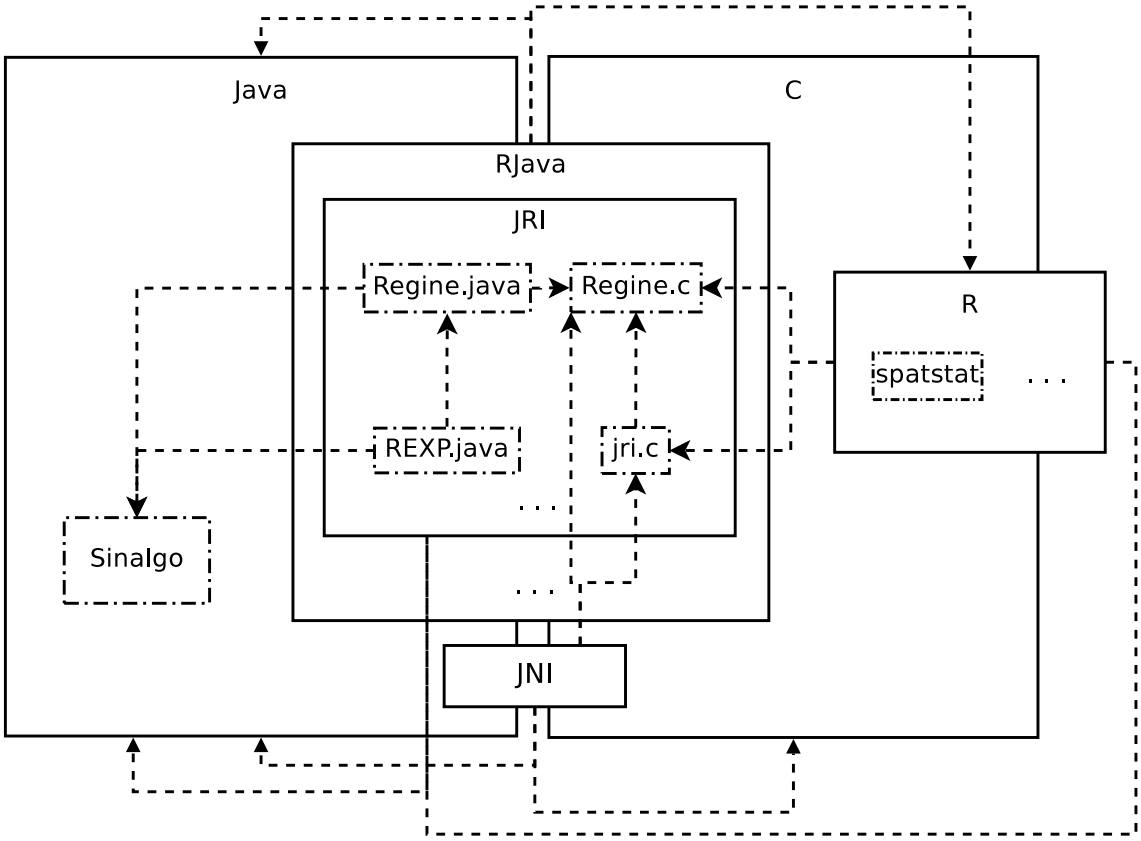


Figura 2. Interação entre Sinalgo e R por meio do RJava

3. Processos Pontuais

Nesta seção descreveremos de forma sumária os principais conceitos dos processos estocásticos pontuais espaciais. Eles são a base teórica para a extensão do simulador Sinalgo. O que segue é baseado no trabalho de [Alencar-Neto 2008].

Os processos pontuais são modelos estocásticos que descrevem a localização de pontos no espaço. Eles são uma poderosa ferramenta para modelar e analisar dados que consistem de pontos distribuídos espacialmente. Eles são aplicados na ecologia vegetal, florestas e silvicultura, na ciência dos materiais, em geografia e sismologia, na astronomia e na epidemiologia, dentre outras [Baddeley 2006]. A plataforma R fornece uma biblioteca completa para a simulação e análise de processos pontuais, o pacote `spatstat` [Baddeley and Turner 2005].

Uma forma conveniente de começar trabalhar com processos pontuais espaciais é através da modificação de um processo básico: o processo de Poisson. Lembremos que a variável aleatória X segue uma distribuição de Poisson com parâmetro $\lambda > 0$ se $\Pr(X = k) = e^{-\lambda}\lambda^k/k!$ para todo k natural. A média de X é λ.

Um número fixo de pontos $n \geq 1$ é dito obedecer um processo pontual Binomial se as coordenadas destes pontos na região retangular $E = [a_1, b_1] \times [a_2, b_2] \subset \mathbb{R}^2$ são resultados de variáveis aleatórias independentes e identicamente distribuídas seguindo uma distribuição uniforme em $[a_i, b_i]$, $i = 1, 2$.

Considerando que o número n de pontos seja definido como o resultado de uma

variável aleatória discreta N que segue a distribuição de Poisson com média $\mu(E)$, podemos definir o processo pontual Poisson da seguinte forma: primeiramente, observa-se n, o resultado da variável aleatória discreta N que segue a distribuição Poisson com intensidade $\eta = \lambda t$ (podemos supor $t = 1$ por simplicidade). Em seguida, posiciona-se os n pontos em E seguindo o processo pontual Binomial definido anteriormente. Então obtem-se como resultado uma ocorrência do processo pontual Poisson homogêneo com *intensidade* $\eta = \lambda$. O processo pontual de Poisson é conhecido como o processo de aleatoriedade completa.

O processo de Matérn *Simple Sequential Inhibition – SSI* é definido iterativamente como o resultado do procedimento que tenta colocar seqüencialmente n pontos sobre a região E, observando uma regra de exclusão com base no parâmetro de exclusão r. A partir do primeiro ponto, enquanto (i) os $n - 1$ pontos restantes não estiverem posicionados ou (ii) o número de tentativas não atingir o limite de iterações T estabelecido, posicionar um novo ponto seguindo a distribuição uniforme; se a distância do novo ponto para qualquer ponto já posicionado for menor que r então este novo ponto será rejeitado e o processo continua gerando um próximo ponto candidato.

Com o processo de Poisson e o processo de Matérn definiremos o processo pontual que iremos incorporar ao Sinalgo: o processo pontual composto $C(n, a)$ que descreve a localização espacial de n pontos através do parâmetro real, a. Sem perda de generalidae, consideremos $E = [0, 100]^2$.

O processo C é construído pela composição de três outros processos pontuais:

1. $B(n, E)$, processo pontual Binomial definido na região E;
2. $M = (n, p, E)$, processo pontual SSI com parâmetro de exclusão $r_{\max}(1 - e^a)$;
3. $S(n, a, \eta, E, E')$, processo pontual composto por dois processos Poisson, um aplicado com intensidade $a\eta$ sobre uma área alvo $E' \subset E$, e outro com intensidade η aplicado sobre a área $E \setminus E'$.

O processo pontual composto $C(n, a)$ fica definido da seguinte forma:

$$\mathcal{C}(n, a) = \begin{cases} M(n, r_{\max}(1 - e^a)) & \text{se } a < 0 \\ B(n) & \text{se } 0 \leq a \leq 1 \\ S(n, a) & \text{se } a > 1. \end{cases} \quad (1)$$

O parâmetro n controla o número de pontos, enquanto o parâmetro a descreve a atratividade com que eles são posicionados na região E. O parâmetro $r_{\max}$ denota o máximo raio de repulsividade possível no modelo, que no nosso caso é $r_{\max} = n^{-1/2}$. Valores negativos da atratividade redundam em um modelo de repulsão de pontos, valores positivos em um modelo onde uma região (E') concentra mais pontos do que o resto do campo, e uma atratividade nula leva ao modelo de aleatoriedade total.

Na próxima seção comentaremos a pertinência deste modelo, e ilustraremos o seu uso para descrever situações práticas de interesse.

4. Resultados: Geração de Topologias

A distribuição espacial dos nós desempenha um papel importante na etapa de simulação em redes de sensores sem fio, uma vez que diversas características de comunicação, como cobertura, conectividade, tempo de vida e qualidade da informação percebida pela

aplicação [Frery et al. 2008, Yau et al. 2008, Wang et al. 2008], irão depender de como os nós estão dispostos na área de interesse.

A despeito da importância da distribuição espacial dos nós para redes de sensores sem fio, a grande maioria dos trabalhos utiliza distribuição totalmente aleatória seguindo um modelo binomial dos nós no campo de sensoreamento [Wang et al. 2008]. Desta maneira, os estudos sobre o comportamento de algoritmos e técnicas empregadas nas redes de sensores são realizados apenas em função dessa distribuição, porém, isto pode não refletir a realidade. [Frery et al. 2008] apresentam diversos cenários de deposição de nós que não são bem representados por distribuições uniformes como, por exemplo, a deposição realizada através de avião em baixas e altas altitudes, a deposição de mais nós em uma determinada região do que em outra (densidades variadas) e a deposição em regiões onde os nós podem ser destruídos como nas proximidades de vulcões, rios etc.

A integração do Sinalgo com o R permite gerar diversos tipos de cenários de topologia com características diferentes das oferecidas pelo software básico. Esses novos cenários aumentam a expressividade da plataforma, permitindo ao projetista analisar situações que antes não podiam ser descritas, conforme detalhado na seção 3.

A seguir apresentamos algumas topologias geradas pelo módulo de geração de topologias proposto neste trabalho. As topologias foram geradas utilizando três diferentes estratégias: (i) modelo atrativo, (ii) modelo aleatório (binomial), e (iii) modelo repulsivo. O parâmetro que controla a atratividade do processo pontual estocástico pode ser modificado pelo usuário através da entrada `Attractivity` encontrada no arquivo `Config.xml` no projeto do Sinalgo. Dessa maneira, o usuário pode controlar o parâmetro de atratividade tanto na interface gráfica quanto em simulações em *batch*.

Empregamos valores de atratividade $a \in [-30, 30]$ na geração dos cenários aqui mostrados. Os valores negativos modelam processos repulsivos, quando $0 \leq a \leq 1$ temos processos binomiais, e quando $a > 1$ temos processos atrativos. Denotamos também $a = -\infty$ como o valor máximo de repulsividade, de maneira que o cenário gerado seja uma grade regular $2D$.

A figura 3 mostra a tela para criação dos nós seguindo o modelo de distribuição SPP (*Stochastic Point Process*). Após a escolha da quantidade de nós, do modelo de distribuição e dos outros modelos utilizados no projeto, os nós serão depostos na área definida para simulação.

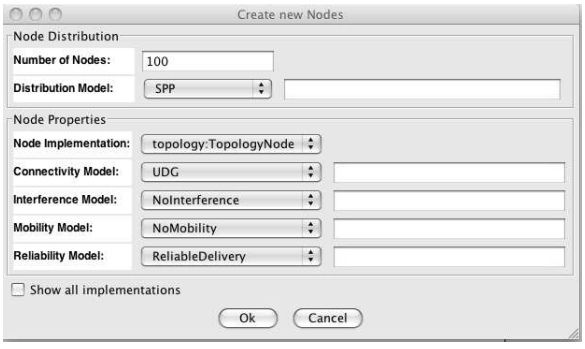
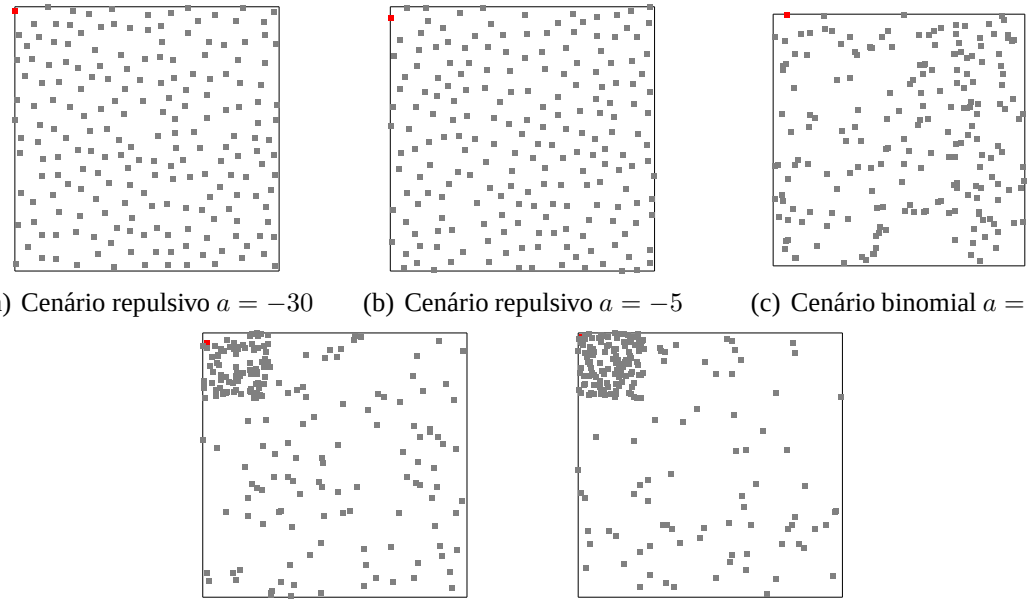


Figura 3. Tela de criação de nós no Sinalgo

A figura 4 apresenta cinco diferentes cenários produzidos pelo modelo proposto



(a) Cenário repulsivo $a = -30$ (b) Cenário repulsivo $a = -5$ (c) Cenário binomial $a = 0$

(d) Cenário atrativo $a = 15$ (e) Cenário atrativo $a = 30$

Figura 4. Diversos cenários de topologia

neste trabalho. Nestas figuras um nó de cor vermelha representa o sorvedouro (*sink*) da rede de sensores e os nós na cor cinza representam os nós sensores. O nó sorvedouro é escolhido como o nó mais próximo à borda superior esquerda da área de sensoriamento. Essas características podem ser facilmente modificadas de modo que o posicionamento do nó sorvedouro seja feito em outras áreas da região de sensoriamento e também pode ser estendido para vários nós sorvedouros.

A figura 4(a) ilustra um cenário de repulsividade com parâmetro $a = -30$. Esse tipo de cenário ilustra os modelos de distribuição de sensores sob condições de controle incompleto, como por exemplo ao lançarmos gradativamente os sensores a partir de um helicóptero sobrevoando num plano de vôo de altura constante. Utilizando este exemplo, quanto mais alto for o plano de vôo, menor será o controle de onde os nós irão ser depositados, por exemplo, na figura 4(b) temos um cenário repulsivo com parâmetro $a = -5$. Percebemos que quando o parâmetro de atratividade cresce até $a = 0$ vamos perdendo o controle na distribuição dos sensores até atingirmos uma deposição totalmente aleatória com distribuição binomial. Este processo pode ser utilizado para modelar o lançamento de sensores através de um helicóptero ou um avião a uma altitude bastante distante da área de sensoriamento. Esta última está apresentada na figura 4(c).

A figura 4(d) apresenta um cenário de atratividade com parâmetro $a = 15$. Esse tipo de cenário também ilustra um modelo de distribuição com controle incompleto mas com objetivos diferentes dos modelos repulsivos. Este modelo representa, por exemplo, a deposição realizada por um helicóptero ou por um avião quando uma distribuição com densidades variadas se faz desejável. Em nossos cenários, foi aumentada a densidade de deposição de nós na região próxima ao sorvedouro. Nas redes de sensores sem fio há uma razão especial para aumentar a densidade de nós nas proximidades do nó sorvedouro pois os nós que se encontram nesta região tendem a consumir mais energia uma vez que eles são responsáveis por transmitir os dados dos nós que estão

mais distantes em direção ao sorvedouro. Esse fato tende a diminuir a vida útil da rede uma vez que quando os nós próximos ao sorvedouro esgotarem suas baterias, nenhuma informação poderá ser entregue com sucesso. Esse efeito é conhecido na literatura por *energy hole* [Li and Mohapatra 2007, Wu et al. 2008]. O modelo atrativo aqui proposto pode ser utilizado com a finalidade de mitigar os efeitos do *energy hole*.

Da mesma maneira que no modelo repulsivo, ao lançarmos os nós sensores de uma altitude mais baixa, teremos mais controle do posicionamento dos nós. A figura 4(e) ilustra um cenário de atratividade com parâmetro $a = 30$. Neste caso percebemos uma alta densidade de nós próximos ao sorvedouro.

Diversos outros cenários típicos das redes de sensores sem fio podem ser imaginados como atrativos ou repulsivos. Por exemplo, na criação de redes esparsas podemos lançar os nós sensores individualmente de modo que eles fiquem distantes uns dos outros. Nesse caso não é plausível que a distribuição de nós fique semelhante à dos processos binomiais como apresentado na figura 4(c). Também não é plausível que sensores lançados em regiões com declive fiquem bem espalhados. Haverá uma clara tendência deles se amontoarem em uma determinada região, o que pode ser modelado por processos atrativos como os apresentados nas figuras 4(d) e 4(e).

5. Conclusões e Trabalhos futuros

A integração entre Sinalgo e R aumenta a capacidade do primeiro de gerar uma quantidade maior de modelos, possibilitando cobrir mais cenários de simulação, e procedimentos para algoritmos desejados. R complementa Sinalgo não somente por apresentar mais recursos estatísticos mas, também, por ter um tratamento de dados confiável, ou seja, há um ganho na precisão numérica que aumenta a confiabilidade dos resultados.

Com R poderemos desenvolver procedimentos para determinados modelos e obter relatórios dos mesmos para serem tratados na sua plataforma, sendo motivação para trabalhos futuros implementar essa volta, ou seja, trazer informações obtidas no Sinalgo para serem tratadas no R.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cyirci, E. (2002). A survey on sensor networks. *Computer Networks*, 38(4):393–422.

Alencar-Neto, J. (2008). Estimação do erro em redes de sensores sem fios. Dissertação de Mestrado em Modelagem Computacional de Conhecimento, Universidade Federal de Alagoas, Maceió, AL.

Almiron, M., Almeida, E. S., and Miranda, M. (2009). The reliability of statistical functions in four software packages freely used in numerical computation. *Brazilian Journal of Probability and Statistics*, Special Issue on Statistical Image and Signal Processing. In press, `http://www.imstat.org/bjps`.

Baddeley, A. (2006). Spatial point processes and their application. In Weil, W., editor, *Stochastic Geometry*, volume 1892 of *Lecture Notes in Mathematics*, pages 1–75. Springer, Berlin.

Baddeley, A. and Turner, R. (2005). spatstat: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42.

Baddeley, A. and Turner, R. (2008). *Package 'spatstat'*, 1.13-0 edition.

Distributed Computing Group (2008). Sinalgo – simulator for network algorithms. Última visita em março de 2009.

Frery, A. C., Ramos, H., Alencar-Neto, J., and Nakamura, E. (2008). Error estimation in wireless sensor networks. In *SAC'08: Proceedings of ACM Symposium on Applied Computing*, volume 3, pages 1927–1932.

Heinzelman, W. B., Chandrakasan, A., and Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communication*, 1:660–670.

Li, J. and Mohapatra, P. (2007). Analytical modeling and mitigation techniques for the energy hole problem in sensor networks. *Pervasive and Mobile Computing*, 3(3):233–254.

Liang, S. (1999). *The Java Native Interface*. Addison-Wesley.

Nakamura, E. F., Loureiro, A. A. F., and Frery, A. C. (2007). Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, 39(3):9/1–55.

R Development Core Team (2008a). *R Data Import/Export*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-10-0.

R Development Core Team (2008b). *R Language Definition*. R Foundation for Statistical Computing, Vienna, Austria. ISBN3-900051-13-5.

R Development Core Team (2008c). *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-11-9.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Reis, I. A., Câmara, G., Assunção, R., and Monteiro, A. M. V. (2007). Data-aware clustering for geosensor networks data collection. In *Anais XIII Simpósio Brasileiro de Sensoriamento Remoto*, pages 6059–6066, Florianópolis, SC, Brazil.

Wang, D. M., Xie, B., and Agrawal, D. P. (2008). Coverage and lifetime optimization of wireless sensor networks with gaussian distribution. *IEEE Transactions on Mobile Computing*, 7(12):1444–1458.

Wu, X., Chen, G., and Das, S. K. (2008). Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Transactions on Parallel and Distributed Systems*, 19(5):710–720.

Yau, D. K. Y., Yip, N. K., Ma, C. Y. T., Rao, N., and Shankar, M. (2008). Quality of monitoring of stochastic events by periodic and proportional-share scheduling of sensor coverage. In *ACM CoNEXT '08: Proceedings of the 4th International Conference on emerging Networking EXperiments and Technologies*.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks*, 52:2292–2330.